

[Python] Pandas 사용법 - DataFrame 생성, 추가, 수정, 삭제, indexing — 나무늘보의 개발 블로그

노트북: 첫 번째 노트북

만든 날짜: 2020-11-02 오전 8:07

URL: <https://continuous-development.tistory.com/131?category=736681>

---

Python

# [Python] Pandas 사용법 - DataFrame 생성, 추가, 수정, 삭제, indexing

2020. 10. 15. 23:31 수정 삭제 공개

## # DataFrame

- 2차원 행렬 데이터에 인덱스를 붙인 것과 동일하다
- 행 인덱스, 열 인덱스를 붙일 수 있다.

## # 데이터 프레임 생성

```
pd.DataFrame(data, index=[], columns=[])
```

*#index 와 columns 는 생략 가능하다*

# 년도에 해당하는 도시별 인구수 정의

```
In [42]: # 년도에 해당하는 도시별 인구수 정의
data = { "2020": [9910293, 8384050, 2938485, 1203948],
         "2018": [8910293, 7384050, 5938485, 3203948],
         "2016": [7910293, 5384050, 7938485, 6203948],
         "2014": [5910293, 3384050, 4938485, 4203948],
         "지역" : ['수도권', '경상권', '수도권', '경상권'],
         "증가율": [0.02343, 0.0434, 0.0944, 0.0034]
       }
pop_df = pd.DataFrame(data, index=['서울', '부산', '경기', '대구'])
pop_df
```

```
Out[42]:
```

	2020	2018	2016	2014	지역	증가율
서울	9910293	8910293	7910293	5910293	수도권	0.02343
부산	8384050	7384050	5384050	3384050	경상권	0.04340
경기	2938485	5938485	7938485	4938485	수도권	0.09440
대구	1203948	3203948	6203948	4203948	경상권	0.00340

data라는 값에 딕셔너리 값을 넣고 이 값을 data 부분에 넣어 데이터 프레임 만들었다. 딕셔너리의 key값은 컬럼명이 되고 들어가는 values는 컬럼 값이 된다.

index 값은 index=[]를 통해 지정해준다.

```
In [44]: # 년도에 해당하는 도시별 인구수 정의
data = { "2020": [9910293, 8384050, 2938485, 1203948],
         "2018": [8910293, 7384050, 5938485, 3203948],
         "2016": [7910293, 5384050, 7938485, 6203948],
         "2014": [5910293, 3384050, 4938485, 4203948],
         "지역" : ['수도권', '경상권', '수도권', '경상권'],
         "증가율": [0.02343, 0.0434, 0.0944, 0.0034]
       }
columns = ["지역", "2014", "2016", "2018", "2020", "증가율"]
pop_df = pd.DataFrame(data, index=['서울', '부산', '경기', '대구'], columns=columns)
pop_df
```

```
Out[44]:
```

	지역	2014	2016	2018	2020	증가율
서울	수도권	5910293	7910293	8910293	9910293	0.02343
부산	경상권	3384050	5384050	7384050	8384050	0.04340
경기	수도권	4938485	7938485	5938485	2938485	0.09440
대구	경상권	4203948	6203948	3203948	1203948	0.00340

이런식으로 처음 DataFrame을 생성할 때 columns 값을 지정해 줄 수도 있다. 이것을 통해 컬럼명을 바꿔줄 수 있다.

```
In [45]: pop_df.columns
```

```
Out[45]: Index(['지역', '2014', '2016', '2018', '2020', '증가율'], dtype='object')
```

컬럼 정보는 해당 데이터 프레임에 **columnse**를 사용해서 볼 수 있다.

```
In [47]: pop_df.index
Out[47]: Index(['서울', '부산', '경기', '대구'], dtype='object')
```

**index** 값은 위와같은 식으로 볼 수 있다.

## #인덱스 이름 / 컬럼 이름 생성

DataFrame.index.name = 인덱스 이름

DataFrame.columns.name = 컬럼이름

```
In [49]: pop_df.index.name = '도시'
          pop_df.columns.name = '특성'
          pop_df
```

```
Out[49]:
```

특성	지역	2014	2016	2018	2020	증가율
도시						
서울	수도권	5910293	7910293	8910293	9910293	0.02343
부산	경상권	3384050	5384050	7384050	8384050	0.04340
경기	수도권	4938485	7938485	5938485	2938485	0.09440
대구	경상권	4203948	6203948	3203948	1203948	0.00340

위와 같은 방식으로 인덱스의 이름과 컬럼의 이름을 지정 해줄 수 있다.

## # random 함수를 통한 DataFrame 생성

데이터 프레임 생성 예제

```
In [67]: from datetime import date,datetime, timedelta

# 정수 컬럼
random_int = np.random.randint(1,100,10)

# 표준 정규분포난수 컬럼
random_gaussian = np.random.randn(10)

#균일 분포 난수 컬럼
random_uniform = np.random.rand(10)

#날짜 컬럼
first_day = datetime(2020,10,10)
days = [first_day + timedelta(day) for day in range(0,10)]

# 문자컬럼
moonja = ['배골','뽕먹지','메뉴','추천종','해주세요','치킨','먹고','싫다','먹으로','갈사람']
```

```
In [68]: data = {
    '정수' : random_int,
    '표준정규분포난수':random_gaussian,
    '균일분포난수':random_uniform,
    '문자':days,
    '날짜':moonja
}
```

```
In [72]: alpha='ABCDEFGHIJ'
columns = ["날짜","문자","정수","표준정규분포난수","균일분포난수"]
phone_df = pd.DataFrame(data, columns=columns,index = [i for i in alpha])
phone_df
```

Out [72]:

	날짜	문자	정수	표준정규분포난수	균일분포난수
A	배골	2020-10-10	33	0.062396	0.587392
B	뽕먹지	2020-10-11	51	0.959119	0.217246
C	메뉴	2020-10-12	59	-0.512734	0.446779
D	추천종	2020-10-13	65	0.685279	0.030035
E	해주세요	2020-10-14	38	0.411600	0.628895
F	치킨	2020-10-15	35	-0.443776	0.438260
G	먹고	2020-10-16	9	-0.567827	0.057674
H	싫다	2020-10-17	61	0.891606	0.186918
I	먹으로	2020-10-18	57	-0.237142	0.752067
J	갈사람	2020-10-19	45	-2.183011	0.803101

## # 컬럼추가

DataFrame['기존에 없던 컬럼명'] = 넣을 값

```
In [74]: pop_df
Out[74]:
```

특성	지역	2014	2016	2018	2020	증가율
도시						
서울	수도권	5910293	7910293	8910293	9910293	0.02343
부산	경상권	3384050	5384050	7384050	8384050	0.04340
경기	수도권	4938485	7938485	5938485	2938485	0.09440
대구	경상권	4203948	6203948	3203948	1203948	0.00340

```
In [79]: pop_df['2014-2016 증가율'] = ((pop_df['2016'] - pop_df['2014']) / pop_df['2014'] * 100).round(2)
In [80]: pop_df
Out[80]:
```

특성	지역	2014	2016	2018	2020	증가율	2014-2016 증가율
도시							
서울	수도권	5910293	7910293	8910293	9910293	0.02343	33.84
부산	경상권	3384050	5384050	7384050	8384050	0.04340	59.10
경기	수도권	4938485	7938485	5938485	2938485	0.09440	60.75
대구	경상권	4203948	6203948	3203948	1203948	0.00340	47.57

해당 DataFrame에 없는 컬럼명을 넣고 값을 추가해서 정의 해주면 컬럼이 추가된다.

## # 컬럼삭제

```
del DataFrame['컬럼명']
```

```
In [81]: # 컬럼 삭제
del pop_df['2014-2016 증가율']
In [82]: pop_df
Out[82]:
```

특성	지역	2014	2016	2018	2020	증가율
도시						
서울	수도권	5910293	7910293	8910293	9910293	0.02343
부산	경상권	3384050	5384050	7384050	8384050	0.04340
경기	수도권	4938485	7938485	5938485	2938485	0.09440
대구	경상권	4203948	6203948	3203948	1203948	0.00340

del 을통해 삭제할 수 있다. 여기서도 배열 인덱싱은 안된다.

## # Series, DataFrame

```
In [84]: type(pop_df['지역'])
Out[84]: pandas.core.series.Series
```

하나를 받을 때는 Series 형으로 받아지고

```
In [85]: type(pop_df[['지역', '증가율']])
Out[85]: pandas.core.frame.DataFrame
```

값이 두개 이상일 때는 DataFrame이 형태로 받아진다.

## # row indexing / slicing

- 인덱스, 문자라벨 슬라이싱도 가능하다.

```
In [94]: pop_df
Out[94]:
```

특성	지역	2014	2016	2018	2020	증가율
도시						
서울	수도권	5910293	7910293	8910293	9910293	0.02343
부산	경상권	3384050	5384050	7384050	8384050	0.04340
경기	수도권	4938485	7938485	5938485	2938485	0.09440
대구	경상권	4203948	6203948	3203948	1203948	0.00340

```
In [105]: pop_df[1:2] # 인덱스 슬라이싱
Out[105]:
```

특성	지역	2014	2016	2018	2020	증가율
도시						
부산	경상권	3384050	5384050	7384050	8384050	0.0434

```
In [106]: pop_df['서울'] # 문자 라벨 슬라이싱
```

```
Out[106]:
```

특성	지역	2014	2016	2018	2020	증가율
도시						
서울	수도권	5910293	7910293	8910293	9910293	0.02343

```
In [108]: pop_df['서울':'경기'] # 문자 라벨 슬라이싱
```

```
Out[108]:
```

특성	지역	2014	2016	2018	2020	증가율
도시						
서울	수도권	5910293	7910293	8910293	9910293	0.02343
부산	경상권	3384050	5384050	7384050	8384050	0.04340
경기	수도권	4938485	7938485	5938485	2938485	0.09440

이런식의 슬라이싱도 가능하다.

## # - 개별 데이터 인덱싱(특정 행, 특정 컬럼)

```
In [111]: pop_df[['서울']][2020]
```

```
Out[111]:
```

도시
서울

Name: 2020, dtype: int64

```
In [112]: pop_df[2020]['서울']
```

```
Out[112]:
```

도시
서울

Name: 2020, dtype: int64

## # 데이터 프레임 예제

- 위 데이터를 보고 모든 학생의 수학점수를 시리즈로 출력하라
- 모든 학생의 국어와 영어 점수를 데이터 프레임으로 만들어라
- 모든 학생의 각 과목 평균 점수를 새로운 열로 추가하라
- 최호진 학생의 영어 점수를 90 점으로 수정하고 평균 점수도 다시 계산하라

- 김지은 학생의 점수를 데이터 프레임으로 만들어라
- 김정수 학생의 점수를 시리즈로 출력하라
- 황인범 학생의 국어점수와 수학점수를 100점으로 수정하고 평균 점수도 다시 계산하라

```

name = exec_df['name']

In [59]: score_data = {
        'kor' : [80,90,70,30],
        'eng' : [90,70,60,40],
        'math' : [90,60,90,70]
    }
    columns = ['kor', 'eng', 'math']
    index = ['김지은', '황인범', '김정수', '최호진']

    exec_df = pd.DataFrame(score_data , index=index, columns=columns)
    exec_df

```

```

In [60]: exec_df['math']

Out[60]: 김지은      90
        황인범      60
        김정수      90
        최호진      70
        Name: math, dtype: int64

```

```

In [61]: exec_df[['kor', 'eng']]

Out[61]:

```

	kor	eng
김지은	80	90
황인범	90	70
김정수	70	60
최호진	30	40



```
In [62]: exec_df['평균'] = (exec_df['math']+exec_df['eng']+exec_df['kor'])/3
```

```
In [63]: exec_df
```

Out [63]:

	kor	eng	math	평균
김지은	80	90	90	86.666667
황인범	90	70	60	73.333333
김정수	70	60	90	73.333333
최호진	30	40	70	46.666667

```
In [72]: exec_df['average'] = np.mean(exec_df[['kor', 'eng', 'math']].T)
display(exec_df)
```

	kor	eng	math	평균	average
김지은	80	90	90	86.666667	86.666667
황인범	90	70	60	73.333333	73.333333
김정수	70	60	90	73.333333	73.333333
최호진	30	90	70	63.333333	63.333333

```
In [64]: exec_df['최호진']['eng'] = 90
```

```
C:\Users\hwang in beom\Anaconda3\lib\site-pack
A value is trying to be set on a copy of a sli
Try using .loc[row_indexer,col_indexer] = valu

See the caveats in the documentation: http://p
"""Entry point for launching an IPython kern
```

```
In [65]: exec_df
```

Out [65]:

	kor	eng	math	평균
김지은	80	90	90	86.666667
황인범	90	70	60	73.333333
김정수	70	60	90	73.333333
최호진	30	90	70	46.666667

```
In [66]: exec_df['평균'] = (exec_df['math']+exec_df['eng']+exec_df['kor'])/3
exec_df
```

Out [66]:

	kor	eng	math	평균
김지은	80	90	90	86.666667
황인범	90	70	60	73.333333
김정수	70	60	90	73.333333
최호진	30	90	70	63.333333

```
In [73]: exec_df['eng']['최호진'] = 90
exec_df['average'] = np.mean(exec_df[['kor', 'eng', 'math']].T)
display(exec_df)
```

C:\Users\hwang in beom\Anaconda3\lib\site-packages\ipykernel\_launcher  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <http://pandas.pydata.org/pandas>  
""Entry point for launching an IPython kernel.

	kor	eng	math	평균	average
김지은	80	90	90	86.666667	86.666667
황인범	90	70	60	73.333333	73.333333
김정수	70	60	90	73.333333	73.333333
최호진	30	90	70	63.333333	63.333333

```
In [67]: print(type(exec_df[:'김지은']))
print(exec_df[:'김지은'])

<class 'pandas.core.frame.DataFrame'>
      kor  eng  math      평균
김지은   80   90   90  86.666667
```

```
In [68]: print(type(exec_df['김정수':'김정수']))  
# print(exec_df[2])
```

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

```
In [70]: print(type(exec_df.loc['김정수']))  
print(exec_df.loc['김정수'])
```

<class 'pandas.core.series.Series'>  
kor 70.000000  
eng 60.000000  
math 90.000000  
평균 73.333333  
Name: 김정수, dtype: float64

```
In [74]: display(exec_df.T['김정수'])
```

kor 70.000000  
eng 60.000000  
math 90.000000  
평균 73.333333  
average 73.333333  
Name: 김정수, dtype: float64

```
In [71]: print(pd.Series([exec_df[2:3]]))  
type(pd.Series([exec_df[2:3]]))
```

0 kor eng math 평균  
김정수 70 60 ...  
dtype: object

Out[71]: pandas.core.series.Series

```
In [137]: exec_df['황인범:']['kor'] = 100
exec_df['황인범:']['math'] = 100
exec_df['평균'] = (exec_df['math']+exec_df['eng']+exec_df['kor'])/3
exec_df
```

C:\Users\whwang in beom\Anaconda3\lib\site-packages\ipykernel\_launcher.py:1: S  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/st>  
"""Entry point for launching an IPython kernel.

C:\Users\whwang in beom\Anaconda3\lib\site-packages\ipykernel\_launcher.py:2: S  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/st>

Out[137]:

	kor	eng	math	평균
김지은	80	90	90	86.666667
황인범	100	70	100	90.000000
김정수	100	60	100	86.666667
최호진	100	90	100	96.666667

```
In [76]: exec_df['average'] = np.mean(exec_df[['kor', 'eng', 'math']].T)
exec_df
```

Out[76]:

	kor	eng	math	평균	average
김지은	80	90	90	86.666667	86.666667
황인범	90	70	60	73.333333	73.333333
김정수	70	60	90	73.333333	73.333333
최호진	30	90	70	63.333333	63.333333

## 'Python' 카테고리의 다른 글

[Python] Pandas 사용법 - 인덱싱 접근, 데이터 조작, 인덱스조작(loc,iloc)

[Python] Pandas 사용법 - 다양한 함수 사용(데이터 입출력, 대소문자변환, 공백...

[Python] Pandas 사용법 - DataFrame 생성, 추가, 수정, 삭제, indexing

[Python] Pandas 사용법 - series 에 대한 추가 , 수정, 삭제, 연산, 결측치□

[Python] Pandas의 이론과 기초적인 사용법□

[Python] Numpy를 통한 난수생성, 카운팅, 통계함수 사용법□

pandas 삭제

pandas 생성

pandas 추가



나무늘보스

혼자 끄적끄적하는 블로그 입니다.