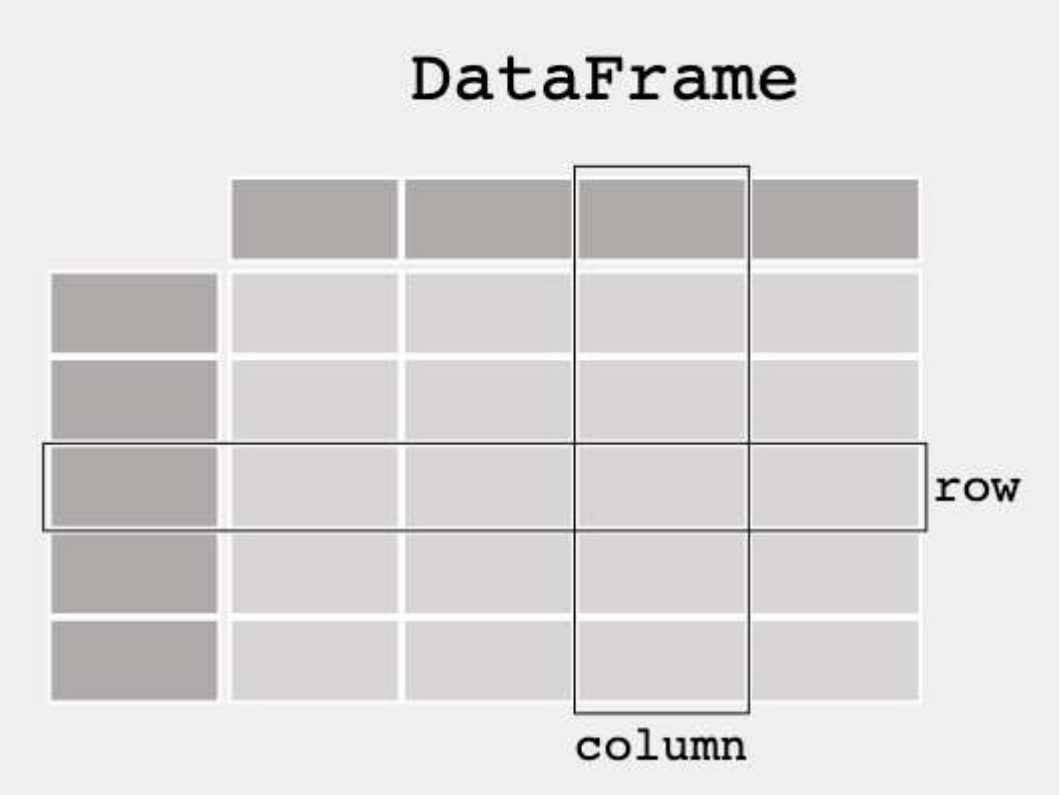
데이터프레임



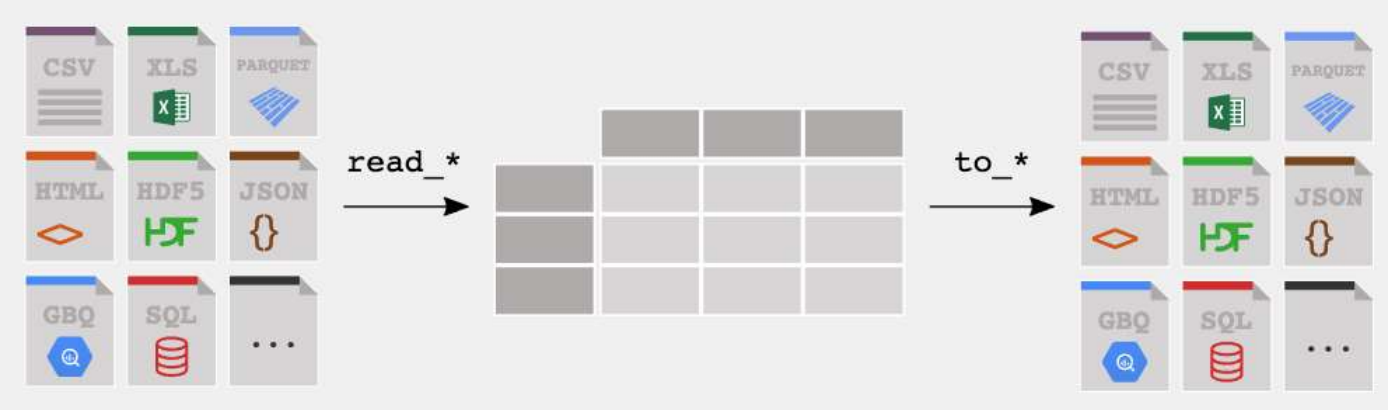
- Pandas 라이브러리에서 기본적으로 데이터를 다루는 단위는 DataFrame : spreadsheet와 같은 개념
- 이러한 형태의 데이터는 Structured Data 또는 Panel Data 또는 Tabular Data라고 부름
- pandas를 공부한다는 것은 결국 dataframe의 사용법을 익히고 활용하는 방법을 배운다는 것과 같더
- pandas를 잘 활용하면 대부분의 structured data를 자유자재로 다룰 수 있게 됨

데이터 프레임

- 2차원 행렬 데이터에 인덱스를 붙인 것
- 행과 열로 만들어지는 2차원 배열 구조
- R의 데이터 프레임 에서 유래
- 데이프레임의 각 열은 시리즈로 구성되어 있음
- DataFrame()함수를 사용해서 생성

외부파일을 데이터 프레임으로 변환 -> 데이터프레임을 외부파일로 변환

- read_* 함수 사용 ex. csv를 데이터프레임으로 변환해서 가져오기 pd.read_csv('파일명')
- to_* 함수사용 ex. 데이터 프레임을 csv 파일로 내보내기 데이터프레임.to_csv('파일명')



데이터 프레임 생성

리스트로 데이터 프레임 만들기

- DataFrame([[list1],[list2]]) - 리스트 안에 리스트 형태로 인수를 전달(2차원 리스트 형태로 전달)
- 각 list는 한 행으로 구성됨
- 행의 원소 개수가 다르면 None 값으로 저장
- index 인수값이 없으면 기본 인덱스 (위치인덱스)가 생성됨

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
df = pd.DataFrame([['a','b','c'],['a','a','g'],['a','a']])
df
```

Out[2]:

	0	1	2
0	a	b	c
1	a	a	g
2	a	a	None

딕셔너리로 데이터프레임 생성

- dict의 key -> column name

In [3]:

```
# 열 데이터를 dict로 작성하는 것이 일반적 임
# 열방향 인덱스(dict의 key), 행방향 인덱스(자동생성 숫자)
df1 = pd.DataFrame(
{
    'A':[90,80,70],
    'B':[85,98,75],
    'C':[88,99,77],
    'D':[87,89,86]
},index = [1,2,3] # 행 인덱스
)
df1
# 딕셔너리의 key는 컬럼네임, index= 인수는 로우네임
```

Out[3]:

	A	B	C	D
1	90	85	88	87
2	80	98	99	89
3	70	75	77	86

In [17]:

```
data = {
    "2015": [9904312, 3448737, 2890451, 2466052],
    "2010": [9631482, 3393191, 2632035, 2000002],
    "2005": [9762546, 3512547, 2517680, 2456016],
    "2000": [9853972, 3655437, 2466338, 2473990],
    "지역": ["수도권", "경상권", "수도권", "경상권"],
    "2010-2015 증가율":[0.0283, 0.0163, 0.0982,0.0141]
}

df3 = pd.DataFrame(data)
df3
```

Out[17]:

	2015	2010	2005	2000	지역	2010-2015 증가율
0	9904312	9631482	9762546	9853972	수도권	0.0283
1	3448737	3393191	3512547	3655437	경상권	0.0163
2	2890451	2632035	2517680	2466338	수도권	0.0982
3	2466052	2000002	2456016	2473990	경상권	0.0141

위에서 생성된 데이터 프레임은 열의 순서가 보장 되지 않고, 각 행의 의미 전달이 어렵다

- index 파라미터와 columns 파라미터를 사용해서 df의 의미 전달이 쉬워진다

In [21]:

```
columns =['지역','2000','2005','2010','2015', '2010-2015 증가율']
index = ['서울','부산','인천','대구']
df3 = pd.DataFrame(data, index=index, columns=columns)
df3
```

Out[21]:

	지역	2000	2005	2010	2015	2010-2015 증가율
서울	수도권	9853972	9762546	9631482	9904312	0.0283
부산	경상권	3655437	3512547	3393191	3448737	0.0163
인천	수도권	2466338	2517680	2632035	2890451	0.0982
대구	경상권	2473990	2456016	2000002	2466052	0.0141

시리즈로 데이터 프레임 생성

- 각 Series의 인덱스 -> columname

In [6]:

```
a = pd.Series([100, 200, 300], ['a', 'b', 'd'])
b = pd.Series([101, 201, 301], ['a', 'b', 'k'])
c = pd.Series([110, 210, 310], ['a', 'b', 'c'])
```

In [10]:

```
pd.DataFrame([a,b,c],index=[100,101,102])
```

Out[10]:

	a	b	d	k	c
100	100.0	200.0	300.0	NaN	NaN
101	101.0	201.0	NaN	301.0	NaN
102	110.0	210.0	NaN	NaN	310.0

csv 데이터로 부터 Dataframe 생성

- 데이터 분석을 위해, dataframe을 생성하는 가장 일반적인 방법
- 데이터 소스로부터 추출된 csv(comma separated values) 파일로부터 생성
- pandas.read_csv 함수 사용

In [7]:

```
# data 출처: https://www.kaggle.com/hesh97/titanicdataset-traincsv/data
train_data = pd.read_csv('./data/train.csv') # 파일경로와 파일명을 정확히 전달해야 함
train_data.head() # df의 위 5행만 출력
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

read_csv 함수 파라미터

- sep - 각 데이터 값을 구별하기 위한 구분자(separator) 설정
- header - header를 무시할 경우, None 설정
- index_col - index로 사용할 column 설정
- usecols - 실제로 dataframe에 로딩할 columns만 설정

In [12]:

```
train_data = pd.read_csv('data/train.csv',
                         index_col = 'PassengerId',
                         usecols= ['PassengerId','Survived','Name','Sex','Age'])
```

In [10]:

```
# df.columns  속성 : df의 컬럼명을 저장하고 있는 속성
train_data.columns
```

Out[10]:

```
Index(['Survived', 'Name', 'Sex', 'Age'], dtype='object')
```

In [11]:

```
train_data.head()
```

Out[11]:

PassengerId	Survived	Name	Sex	Age
1	0	Braund, Mr. Owen Harris	male	22.0
2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0
3	1	Heikkinen, Miss. Laina	female	26.0
4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0
5	0	Allen, Mr. William Henry	male	35.0

인덱스와 컬럼의 이해

1. 인덱스(index)
   - index 속성
   - 각 아이템을 특정할 수 있는 고유의 값을 저장
   - 복잡한 데이터의 경우, 멀티 인덱스로 표현 가능

2. 컬럼(column)
   - columns 속성
   - 각각의 특성(feature)을 나타냄
   - 복잡한 데이터의 경우, 멀티 컬럼으로 표현 가능

In [20]:

```
df3
```

Out[20]:

	2015	2010	2005	2000	지역	2010-2015 증가율
0	9904312	9631482	9762546	9853972	수도권	0.0283
1	3448737	3393191	3512547	3655437	경상권	0.0163
2	2890451	2632035	2517680	2466338	수도권	0.0982
3	2466052	2000002	2456016	2473990	경상권	0.0141

In [18]:

```
# df의 컬럼명(열인덱스)을 확인 - df.columns 속성
print(df3.columns)
type(df3.columns)
```

```
Index(['2015', '2010', '2005', '2000', '지역', '2010-2015 증가율'], dtype='object')
```

Out[18]:

```
pandas.core.indexes.base.Index
```

In [22]:

```
#df의 행 인덱스를 확인 - df.index 속성
print(type(df3.index))
df3.index
```

```
<class 'pandas.core.indexes.base.Index'>
```

Out[22]:

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

행/열 인덱스 이름 설정

- index.name
- columns.name

In [23]:

```
df3.index.name ='도시'
df3.columns.name = '특성'
df3
```

Out[23]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	0.0283
부산	경상권	3655437	3512547	3393191	3448737	0.0163
인천	수도권	2466338	2517680	2632035	2890451	0.0982
대구	경상권	2473990	2456016	2000002	2466052	0.0141

In [25]:

```
# 데이터 프레임내의 데이터만 접근하려면 values 속성을 사용
print(type(df3.values))
df3.values # df의 value는 2차원 ndarray
```

```
<class 'numpy.ndarray'>
```

Out[25]:

```
array([['수도권', 9853972, 9762546, 9631482, 9904312, 0.0283],
       ['경상권', 3655437, 3512547, 3393191, 3448737, 0.0163],
       ['수도권', 2466338, 2517680, 2632035, 2890451, 0.0982],
       ['경상권', 2473990, 2456016, 2000002, 2466052, 0.0141]], dtype=object)
```

In [27]:

```
type(df3.values[0])
df3.values[0]
```

Out[27]:

```
array(['수도권', 9853972, 9762546, 9631482, 9904312, 0.0283], dtype=object)
```

dataframe 데이터 파악하기

- shape 속성 (row, column)
- describe 함수 - 숫자형 데이터의 통계치 계산
- info 함수 - 데이터 타입, 각 아이템의 개수 등 출력

In [32]:

```
train_data.head()
```

Out[32]:

PassengerId	Survived	Name	Sex	Age
1	0	Braund, Mr. Owen Harris	male	22.0
2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0
3	1	Heikkinen, Miss. Laina	female	26.0
4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0
5	0	Allen, Mr. William Henry	male	35.0

In [30]:

```
len(train_data) # df의행수
print(train_data.size) # df의 값의 개수
train_data.shape # df의 행과 열 수
```

```
3564
```

Out[30]:

```
(891, 4)
```

In [31]:

```
# 데이터의 요약(개요)정보 반환
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Name      891 non-null    object 
 2   Sex       891 non-null    object 
 3   Age       714 non-null    float64
dtypes: float64(1), int64(1), object(2)
memory usage: 34.8+ KB
```

In [33]:

```
train_data.describe() # 수치형 데이터에 대해서만 기본 통계량을 반환
```

Out[33]:

	Survived	Age
count	891.000000	714.000000
mean	0.383838	29.699118
std	0.486592	14.526497
min	0.000000	0.420000
25%	0.000000	20.125000
50%	0.000000	28.000000
75%	1.000000	38.000000
max	1.000000	80.000000

데이터 프레임 전치

- 판다스 데이터 프레임은 전치를 포함해서 Numpy 2차원 배열의 대부분 속성이나 메서드를 지원함.
- 전치 : 행과 열을 바꾸는 기능
- df.T

In [34]:

```
df3
```

```
...
```

In [35]:

```
# df를 전치한 결과를 반환 - 원본 df는 변경되지 않음
df3.T
```

Out[35]:

도시	서울	부산	인천	대구
특성				
지역	수도권	경상권	수도권	경상권
2000	9853972	3655437	2466338	2473990
2005	9762546	3512547	2517680	2456016
2010	9631482	3393191	2632035	2000002
2015	9904312	3448737	2890451	2466052
2010-2015 증가율	0.0283	0.0163	0.0982	0.0141

In [50]:

```
# 확인 후 설명
df3.T['서울'].values
df3.T['서울']['2000']
```

Out[50]:

```
array(['수도권', 9853972, 9762546, 9631482, 9904312, 0.0283], dtype=object)
```

데이터 프레임 내용 변경 :

- 열추가, 열삭제, 내용 갱신

In [51]:

```
df3
```

Out[51]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	0.0283
부산	경상권	3655437	3512547	3393191	3448737	0.0163
인천	수도권	2466338	2517680	2632035	2890451	0.0982
대구	경상권	2473990	2456016	2000002	2466052	0.0141

해당열이 있으면 내용 갱신, 열이 없으면 추가

- 열추가 : df[열이름(key)]=values

- 열 내용 갱신 : df[열이름(key)]=values

In [52]:

```
df3.columns
```

Out[52]:

```
Index(['지역', '2000', '2005', '2010', '2015', '2010-2015 증가율'], dtype='object',
name='특성')
```

In [53]:

```
df3['2010-2015 증가율'] * 100
```

Out[53]:

```
도시
서울     2.83
부산     1.63
인천     9.82
대구     1.41
Name: 2010-2015 증가율, dtype: float64
```

In [54]:

```
# 열 데이터 갱신
df3['2010-2015 증가율'] = df3['2010-2015 증가율'] * 100
```

In [55]:

```
df3['2010-2015 증가율']
```

Out[55]:

```
도시
서울     2.83
부산     1.63
인천     9.82
대구     1.41
Name: 2010-2015 증가율, dtype: float64
```

In [59]:

```
# 열 추가
df3['비고'] = ['특별시','광역시','특례시','특례시']
df3
```

Out[59]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율	비고
도시							
서울	수도권	9853972	9762546	9631482	9904312	2.83	특별시
부산	경상권	3655437	3512547	3393191	3448737	1.63	광역시
인천	수도권	2466338	2517680	2632035	2890451	9.82	특례시
대구	경상권	2473990	2456016	2000002	2466052	1.41	특례시

In [60]:

```
# 열삭제 : del df['삭제열'] - 완전 삭제됨(원본반영됨) 두번이상 실행 시 에러
del df3['비고']
df3
```

Out[60]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82
대구	경상권	2473990	2456016	2000002	2466052	1.41

In [61]:

```
# 가공열 추가
df3['2005-2015 증가율'] = ((df3['2015']-df3['2005'])/df3['2005']*100).round(2)
df3
```

Out[61]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율	2005-2015 증가율
도시							
서울	수도권	9853972	9762546	9631482	9904312	2.83	1.45
부산	경상권	3655437	3512547	3393191	3448737	1.63	-1.82
인천	수도권	2466338	2517680	2632035	2890451	9.82	14.81
대구	경상권	2473990	2456016	2000002	2466052	1.41	0.41

In [62]:

```
del df3['2005-2015 증가율']
df3
```

Out[62]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82
대구	경상권	2473990	2456016	2000002	2466052	1.41

DF의 행추가

- pd의 인덱서 사용

- concate() 사용 - 추가하고자 하는 data를 df로 새로 생성후 결합

In [63]:

```
df3
```

Out[63]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82
대구	경상권	2473990	2456016	2000002	2466052	1.41

In [64]:

```
# loc 인덱서 사용 : df.loc[새로추가될행이름]=[데이터1,...,데이터n]
df3.loc['광주']=['호남권',2470000,2456000,2453000,2460000,1.00]
```

In [65]:

```
df3
```

Out[65]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82
대구	경상권	2473990	2456016	2000002	2466052	1.41
광주	호남권	2470000	2456000	2453000	2460000	1.00

데이터 프레임 인덱싱

1. 열인덱싱
2. 인덱서를 사용하지않는 행 인덱싱

- []기호를 이용해서 인덱싱할때 주의점 : []기호는 열 위주 인덱싱이 원칙

1. 열인덱싱

1.열 라벨(컬럼명)을 키값으로 생각하고 인덱싱한다.

- 인덱스로 라벨값을 하나 넣으면 시리즈 객체가 반환
- 라벨의 배열이나 리스트를 넣으면 부분적 df 가 반환

In [66]:

```
df3
```

Out[66]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82
대구	경상권	2473990	2456016	2000002	2466052	1.41
광주	호남권	2470000	2456000	2453000	2460000	1.00

한개의 열 추출

In [68]:

```
# 인덱스로 키워드 사용 -  시리즈 형태로 반환
print(df3['지역'])
type(df3['지역'])
```

```
도시
서울    수도권
부산    경상권
인천    수도권
대구    경상권
광주    호남권
Name: 지역, dtype: object
```

Out[68]:

```
pandas.core.series.Series
```

In [69]:

```
# 열 1개 추출시 . 연산자 사용 가능(시리즈로 반환)
df3.지역
```

Out[69]:

```
도시
서울    수도권
부산    경상권
인천    수도권
대구    경상권
광주    호남권
Name: 지역, dtype: object
```

In [70]:

```
# 인덱스 값으로 컬럼명의 리스트를 사용하면 반환되는 데이터는 DF
df3[['지역']]
```

Out[70]:

특성	지역
도시	
서울	수도권
부산	경상권
인천	수도권
대구	경상권
광주	호남권

In [71]:

```
type(df3[['지역']])
```

Out[71]:

```
pandas.core.frame.DataFrame
```

여러개의 열 추출

- 인덱스 값으로 list 사용
- DF로 반환

In [73]:

```
df3[['2010','2015']]
```

Out[73]:

특성	2010	2015
도시		
서울	9631482	9904312
부산	3393191	3448737
인천	2632035	2890451
대구	2000002	2466052
광주	2453000	2460000

판다스 데이터 프레임에 열이름(컬럼명)이 문자열일 경우에는

- 수치 인덱스를 사용할 수 없음
- 위치 인덱싱 기능을 사용할 수 없다. : keyerror 발생

In [74]:

```
# 위치적으로 맨 처음 열을 반환받기 위해 위치 인덱스 사용해 봄
try :
    df3[0] # <class 'KeyError'>
except Exception as e :
    print(type(e))

# df3[0]은 컬럼명이 0인 컬럼을 찾으라는 의미인데
# df3에는 컬럼명이 0인 컬럼은 없음 - 위치 인덱스 사용 불가
```

```
<class 'KeyError'>
```

- 위치 인덱싱처럼 보이는 예제

In [75]:

```
# 예제 df5 생성
df5 = pd.DataFrame(np.arange(12).reshape(3,4))
# df 만들때 index와 column을 명시하지 않음
# 이런경우 기본 인덱스가 자동 생성됨(0부터시작하는 인덱스가 생성)
# 순서(위치)가 아님(위치랑은 아무 상관 없음)

df5
```

Out[75]:

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11

In [77]:

```
df5[[1,2]]
# 위치인덱스가 아니고 컬럼명이 숫자로 되어있는 경우임
```

Out[77]:

	1	2
0	1	2
1	5	6
2	9	10

In [79]:

```
np.arange(12)
np.arange(12).reshape(3,4) # 배열의 차원을 변경
```

Out[79]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

행 단위 인덱싱

- 행단위 인덱싱을 하고자 하면 인덱서라는 특수 기능을 사용하지 않는 경우 슬라이싱을 해야 함(인덱서는 바로 뒤에 배움)
- 인덱스 값이 문자(라벨)면 문자슬라이싱도 가능하다.
  - 위치값 슬라이싱도 가능

In [80]:

```
df3
```

Out[80]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82
대구	경상권	2473990	2456016	2000002	2466052	1.41
광주	호남권	2470000	2456000	2453000	2460000	1.00

In [82]:

```
# 첫번째 행 추출
df3[:'서울']
```

Out[82]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83

In [83]:

```
df3['인천':'인천']
```

Out[83]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
인천	수도권	2466338	2517680	2632035	2890451	9.82

In [85]:

```
df3[:1]
df3[0:1]
```

Out[85]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83

In [86]:

```
df3[1:3] # [시작값 : 끝값+1]
```

Out[86]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82

In [87]:

```
# 키워드 인덱스를 이용
df3['부산':'인천']
```

Out[87]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82

In [88]:

```
df3['광주':'광주']
```

Out[88]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
광주	호남권	2470000	2456000	2453000	2460000	1.0

- 개별요소 접근 [열][행]

In [89]:

```
df3['2015'] # 시리즈 반환
```

Out[89]:

```
도시
서울    9904312
부산    3448737
인천    2890451
대구    2466052
광주    2460000
Name: 2015, dtype: int64
```

In [90]:

```
df3['2015']['부산']
# 2010 열을 시리즈로 우선 반환받고 시리즈에서 부산 인덱스에 해당하는 값을 반환
```

Out[90]:

```
3448737
```

In [91]:

```
df3[['2015']] # df로 반환
```

Out[91]:

특성	2015
도시	
서울	9904312
부산	3448737
인천	2890451
대구	2466052
광주	2460000

In [93]:

```
# df3[['2015']]['부산'] # key에러 발생
# df3[['2015']] 데이터 프레임을 반환하기 때문
```

In [94]:

```
df3[['2015']] ['부산':'부산'] # df 반환
```

Out[94]:

특성	2015
도시	
부산	3448737

In [96]:

```
df = df3[['2005','2010']]
df
```

Out[96]:

특성	2005	2010
도시		
서울	9762546	9631482
부산	3512547	3393191
인천	2517680	2632035
대구	2456016	2000002
광주	2456000	2453000

In [97]:

```
# 부산데이터 추출
df3[['2005','2010']]['부산':'부산']
```

Out[97]:

특성	2005	2010
도시		
부산	3512547	3393191

In [99]:

```
df3.head()
```

Out[99]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82
대구	경상권	2473990	2456016	2000002	2466052	1.41
광주	호남권	2470000	2456000	2453000	2460000	1.00

In [103]:

```
### 열 인덱싱에 슬라이싱 사용 - 사용불가
# df3[['2000':'2010']]
```

행 삭제

- drop() 함수 사용
- drop(index=[삭제할 행 인덱스])
- 원본반영하지 않는다.
- 삭제와 동시에 원본 반영하려면
  - drop(index=[삭제할 행 인덱스], inplace=True)

In [104]:

```
df3
```

Out[104]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82
대구	경상권	2473990	2456016	2000002	2466052	1.41
광주	호남권	2470000	2456000	2453000	2460000	1.00

In [105]:

```
df3.drop(index=['광주'])
```

Out[105]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82
대구	경상권	2473990	2456016	2000002	2466052	1.41

In [106]:

```
df3
```

Out[106]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82
대구	경상권	2473990	2456016	2000002	2466052	1.41
광주	호남권	2470000	2456000	2453000	2460000	1.00

In [107]:

```
df3.drop(index=['광주','대구'])
```

Out[107]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82

In [108]:

```
# 원본 수정
df3.drop(index=['광주'],inplace=True)
```

In [109]:

```
df3
```

Out[109]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82
대구	경상권	2473990	2456016	2000002	2466052	1.41

drop() 이용 열 삭제

- drop(columns=[삭제할 열])

In [111]:

```
df3.drop(columns=['2010-2015 증가율','2010'])
```

Out[111]:

특성	지역	2000	2005	2015
도시				
서울	수도권	9853972	9762546	9904312
부산	경상권	3655437	3512547	3448737
인천	수도권	2466338	2517680	2890451
대구	경상권	2473990	2456016	2466052

drop()함수 columns/index 미 표기시

- drop([삭제할행 또는 열],axis=0/1)
- axis : 0(행),1(열)

In [112]:

```
df3.drop(['대구'],axis=0)
```

Out[112]:

특성	지역	2000	2005	2010	2015	2010-2015 증가율
도시						
서울	수도권	9853972	9762546	9631482	9904312	2.83
부산	경상권	3655437	3512547	3393191	3448737	1.63
인천	수도권	2466338	2517680	2632035	2890451	9.82

In [113]:

```
df3.drop(['2010-2015 증가율'],axis=1)
```

Out[113]:

특성	지역	2000	2005	2010	2015
도시					
서울	수도권	9853972	9762546	9631482	9904312
부산	경상권	3655437	3512547	3393191	3448737
인천	수도권	2466338	2517680	2632035	2890451
대구	경상권	2473990	2456016	2000002	2466052