Data Frame

In [4]:
```
import numpy as np
import pandas as pd
```

1. 하나의 열이 되는 data를 list나 1차원 배열로 준비

2. 각 열에 대한 이름(label)을 key로 가지는 dictionary를 만듦

In [1]:
```
data = {
    "2015": [9904312, 3448737, 2890451, 2466052],
    "2010": [9631482, 3393191, 2632035, 2431774],
    "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]
}
data
```

Out[1]:
```
{'2015': [9904312, 3448737, 2890451, 2466052],
 '2010': [9631482, 3393191, 2632035, 2431774],
 '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]}
```

3. data를 Data Frame 생성자에 넣음,

(열 index = column 인수/행 인덱스 = index 인수)

In [9]:
```
columns = ["지역", "2015", "2010", "2005", "2000", "2010-2015 증가율"]
index = ["서울", "부산", "인천", "대구"]
df = pd.DataFrame(data, index=index, columns=columns)
df
```

Out[9]:

	지역	2015	2010	2005	2000	2010-2015 증가율
서울	수도권	9904312	9631482	9762546	9853972	0.0283
부산	경상권	3448737	3393191	3512547	3655437	0.0163
인천	수도권	2890451	2632035	2517680	2466338	0.0982
대구	경상권	2466052	2431774	2456016	2473990	0.0141

In [10]:
```
df.values
```

Out[10]:
```
array([['수도권', 9904312, 9631482, 9762546, 9853972, 0.0283],
       ['경상권', 3448737, 3393191, 3512547, 3655437, 0.0163],
       ['수도권', 2890451, 2632035, 2517680, 2466338, 0.0982],
       ['경상권', 2466052, 2431774, 2456016, 2473990, 0.0141]], dtype=object)
```

In [15]:
```
df.index
```

Out[15]:
```
Index(['서울', '부산', '인천', '대구'], dtype='object')
```

In [14]:
```
df.columns
```

Out[14]:
```
Index(['지역', '2015', '2010', '2005', '2000', '2010-2015 증가율'], dtype='object')
```

In [16]:
```
df["2015"]          # ["label"]: Series return
```

Out[16]:
```
서울    9904312
부산    3448737
인천    2890451
대구    2466052
Name: 2015, dtype: int64
```

In [23]:
```
df[["2015"]]          # [["label"]]: DataFrame return
```

Out[23]:

	2015
서울	9904312
부산	3448737
인천	2890451
대구	2466052

In [22]:
```
df[["2010", "2015"]]  # ["label1", "label2", ...]: DataFrame return
```

Out[22]:

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

In [17]:
```
df["2015"].values
```

Out[17]:
```
array([9904312, 3448737, 2890451, 2466052], dtype=int64)
```

In [18]:
```
df["2010-2015 증가율"] = df["2010-2015 증가율"] * 100
df
```

Out[18]:

	지역	2015	2010	2005	2000	2010-2015 증가율
서울	수도권	9904312	9631482	9762546	9853972	2.83
부산	경상권	3448737	3393191	3512547	3655437	1.63
인천	수도권	2890451	2632035	2517680	2466338	9.82
대구	경상권	2466052	2431774	2456016	2473990	1.41

In [21]:
```
# round(소수점 자리수) 반올림함수
df["2005-2010 증가율"] = ((df["2010"]-df["2005"])/df["2005"]*100).round(2)
df
```

Out[21]:

	지역	2015	2010	2005	2000	2010-2015 증가율	2005-2010 증가율
서울	수도권	9904312	9631482	9762546	9853972	2.83	-1.34
부산	경상권	3448737	3393191	3512547	3655437	1.63	-3.40
인천	수도권	2890451	2632035	2517680	2466338	9.82	4.54
대구	경상권	2466052	2431774	2456016	2473990	1.41	-0.99

In [27]:
```
# 행 indexing -> slicing
df[1:2]
```

Out[27]:

	지역	2015	2010	2005	2000	2010-2015 증가율	2005-2010 증가율
부산	경상권	3448737	3393191	3512547	3655437	1.63	-3.4

In [28]:
```
df["서울":"부산"]
```

Out[28]:

	지역	2015	2010	2005	2000	2010-2015 증가율	2005-2010 증가율
서울	수도권	9904312	9631482	9762546	9853972	2.83	-1.34

	지역	2015	2010	2005	2000	2010-2015 증가율	2005-2010 증가율
부산	경상권	3448737	3393191	3512547	3655437	1.63	-3.40

In [31]:
```
# df["열"]["행"]
df["2015"]["부산"]
```

Out[31]: 3448737

문제 1

In [7]:
```
data_subject = {
    "국어": [80,90,70,30],
    "영어": [90,70,60,40],
    "수학": [90,60,80,70],
    "이름": ["춘향", "몽룡", "향단", "방자"]
}

columns_subject = ["국어", "영어", "수학"]
index_subject = ["춘향", "몽룡", "향단", "방자"]
df_subject = pd.DataFrame(data_subject, index = index_subject, columns = columns_subject)
df_subject
```

Out[7]:

	국어	영어	수학
춘향	80	90	90
몽룡	90	70	60
향단	70	60	80
방자	30	40	70

In [32]:
```
df_subject["수학"] # 수학점수를 Series로
```

Out[32]:
```
춘향    90
몽룡    60
향단    80
방자    70
Name: 수학, dtype: int64
```

In [36]:
```
df_subject[["국어", "영어"]] # 국어,영어 점수를 Data Frame으로
```

Out[36]:

	국어	영어
춘향	80	90
몽룡	90	70
향단	70	60
방자	30	40

In [39]:
```
df_subject["average"] = ((df_subject["국어"]+df_subject["영어"]+df_subject["수학"])/3).round(2)
df_subject
```

Out[39]:

	국어	영어	수학	average
춘향	80	90	90	86.67
몽룡	90	70	60	73.33
향단	70	60	80	70.00
방자	30	40	70	46.67

In [42]:
```
df_subject["영어"]["방자"]=80
```

```
<ipython-input-42-92787d4725cd>:1: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  df_subject["영어"]["방자"]=80
```

In [43]:
```
df_subject["average"] = ((df_subject["국어"]+df_subject["영어"]+df_subject["수학"])/3).round(2)
df_subject
```

Out[43]:

	국어	영어	수학	average
춘향	80	90	90	86.67
몽룡	90	70	60	73.33
향단	70	60	80	70.00
방자	30	80	70	60.00

In [49]:
```
df_subject[:1] # 행 indexing으로 춘향 dataframe
```

Out[49]:

	국어	영어	수학	average
춘향	80	90	90	86.67

DataFrame 고급 인덱싱

In [52]:
```
df_alphabet = pd.DataFrame(np.arange(10,22).reshape(3,4),
                           index=["a", "b", "c"],
                           columns=["A","B","C","D"])
df_alphabet
```

Out[52]:

	A	B	C	D
a	10	11	12	13
b	14	15	16	17
c	18	19	20	21

loc: 라벨값 기반의 2차원 indexing

In [72]:
```
df_alphabet.loc['a']
```

Out[72]:
```
A    10
B    11
C    12
D    13
Name: a, dtype: int32
```

In [87]:
```
df_alphabet.loc['b':'c'] # == df_alphabet['b':'c'] == df_alphabet[["b","c"]]
```

Out[87]:

	A	B	C	D
b	14	15	16	17
c	288	304	320	336

In [88]:
```
df_alphabet.A > 15
```

Out[88]:
```
a    False
b    False
c     True
Name: A, dtype: bool
```

In [89]:
```
df_alphabet.loc[df_alphabet.A > 15]
```

Out[89]:

	A	B	C	D

	A	B	C	D
c	288	304	320	336

In [92]:
```
def select_rows(df):
    return df.A > 15
df_alphabet.loc[select_rows(df_alphabet)]
```

Out[92]:

	A	B	C	D
c	288	304	320	336

In [94]:
```
df_alphabet.loc[:, 'A']
```

Out[94]:
```
a     10
b     14
c    288
Name: A, dtype: int32
```

In [96]:
```
df_alphabet.loc["b":, "A"]
```

Out[96]:
```
b     14
c    288
Name: A, dtype: int32
```

In [97]:
```
df_alphabet.loc["a", :]
```

Out[97]:
```
A    10
B    11
C    12
D    13
Name: a, dtype: int32
```

In [101]:
```
df_alphabet
```

Out[101]:

	A	B	C	D
a	10	11	12	13
b	14	15	16	17
c	288	304	320	336

In [103]:
```
df_alphabet.loc[df_alphabet.A > 10, ["C", "D"]]
```

Out[103]:

	C	D
b	16	17
c	320	336

iloc: 정수 기반의 2차원 indexing

In [57]:
```
df_alphabet.iloc[0,1]
```

Out[57]:
```
11
```

In [58]:
```
df_alphabet.iloc[:2,2]
```

Out[58]:
```
a    12
b    16
Name: C, dtype: int32
```

In [61]:
```
df_alphabet.iloc[0,-2:]
```

Out[61]:
```
C    12
D    13
Name: a, dtype: int32
```

In [62]:
```
df_alphabet.iloc[2:3, 1:3]
```

Out[62]:

	B	C
c	19	20

In [70]:
```
df_alphabet.iloc[-1] = df_alphabet.iloc[-1]*2
df_alphabet
```

Out[70]:

	A	B	C	D
a	10	11	12	13
b	14	15	16	17
c	288	304	320	336

DataFrame - 개수 세기

In [73]:
```
s = pd.Series(range(10))
s[3]=np.nan
s
```

Out[73]:
```
0    0.0
1    1.0
2    2.0
3    NaN
4    4.0
5    5.0
6    6.0
7    7.0
8    8.0
9    9.0
dtype: float64
```

In [74]:
```
s.count() #NaN 값은 빼고 갯수 셈
```

Out[74]:
```
9
```

In [78]:
```
np.random.seed(2)
df = pd.DataFrame(np.random.randint(5,size=(4,4)), dtype=float) # 0~5사이의 숫자
df.iloc[2,3] = np.nan
df
```

Out[78]:

	0	1	2	3
0	0.0	0.0	3.0	2.0
1	3.0	0.0	2.0	1.0
2	3.0	2.0	4.0	NaN
3	4.0	3.0	4.0	2.0

In [79]:
```
df.count() # 각 열마다 데이터 개수 카운팅
```

Out[79]:
```
0    4
1    4
2    4
3    3
dtype: int64
```

DataFrame - 정렬

In [80]:
```
df_subject.sort_values(by='국어')
```

Out[80]:

	국어	영어	수학	average

	국어	영어	수학	average
방자	30	80	70	60.00
향단	70	60	80	70.00
춘향	80	90	90	86.67
몽룡	90	70	60	73.33

In [81]:
```
df_subject.sort_values(by=['국어','수학']) # 국어로 sorting-> 동점이 있으면 수학으로 sorting
```

Out[81]:

	국어	영어	수학	average
방자	30	80	70	60.00
향단	70	60	80	70.00
춘향	80	90	90	86.67
몽룡	90	70	60	73.33

행 / 열 합계

In [82]:
```
df_subject.sum(axis=1) # 행 합산
```

Out[82]:
```
춘향    346.67
몽룡    293.33
향단    280.00
방자    240.00
dtype: float64
```

In [86]:
```
df_subject["RowSum"]=df_subject.sum(axis=1)
df_subject
```

Out[86]:

	국어	영어	수학	average	RowSum
춘향	80.0	90.0	90.0	86.67	346.67
몽룡	90.0	70.0	60.0	73.33	293.33
향단	70.0	60.0	80.0	70.00	280.00
방자	30.0	80.0	70.0	60.00	240.00
ColTotal	540.0	600.0	600.0	580.00	2320.00

In [83]:
```
df_subject.sum() # 열 합산
```

Out[83]:
```
국어          270.0
영어          300.0
수학          300.0
average     290.0
dtype: float64
```

In [85]:
```
df_subject.loc["ColTotal", :] = df_subject.sum()
df_subject
```

Out[85]:

	국어	영어	수학	average
춘향	80.0	90.0	90.0	86.67
몽룡	90.0	70.0	60.0	73.33
향단	70.0	60.0	80.0	70.00
방자	30.0	80.0	70.0	60.00
ColTotal	540.0	600.0	600.0	580.00