
▣ Ch04 데이터 프레임(Data Frame) 자료 다루기

(Ch16과 Ch04 내용을 함께 구성한다.)

- > 외장 **Pandas** 자료형
- > **pandas**라는 패키지를 **import** 해서 사용 가능한 추가 자료 구조
- > Data Type : **series, dataframe**
- > 관련 함수 : **Series(), DataFrame()**

○ 시리즈(Series) 형

- > 1차원 배열로 여러 값을 나열한 자료 구조
- > 각 데이터 항목은 **index**로 식별된다.
- > 데이터 프레임의 데이터를 구성하는 열 값을 시리즈로 추출하여 조작할 수 있다.
- > **pandas** 패키지를 통해 사용할 수 있다.

```
In [1]: ## 시리즈(Series) 형 ##
## 생성
import pandas as pd
a = pd.Series([11, 22, 33, 44], index = [1, 2, 3, 4]) #index 강제 부여
a
```

```
Out[1]: 1    11
2    22
3    33
4    44
dtype: int64
```

○ 데이터 프레임(Data Frame) 형

- > 행과 열로 나열되는 2차원적 자료 구조
- > 행과 열 값들은 딕셔너리 구조를 활용하여 초기화 한다.
- > 하나의 열 값들은 시리즈형으로 추출하여 사용 가능하다.
- > 복수 개의 열 값들은 데이터 프레임형으로 추출하여 사용 가능하다.
- > **pandas** 패키지를 통해 사용할 수 있다.

[] 데이터 프레임 생성

```
In [84]: ## [Data Frame] 생성
import pandas as pd
df = pd.DataFrame({'Class' : [1, 1, 2],
                   'Eng' : [87, 79, 80],
                   'Mat' : [80, 90, 80]}) #index 자동 부여
df
```

```
Out[84]:   Class  Eng  Mat
0      1    87    80
1      1    79    90
2      2    80    80
```

```
In [99]: ## [Data Frame] 생성
import pandas as pd
df = pd.DataFrame({'Class' : [1, 1, 2],
                   'Eng' : [87, 79, 80],
                   'Mat' : [80, 90, 80]},
                   index = ['Kims', 'Lees', 'Parks']) #index 강제 부여
df
```

```
Out[99]:   Class  Eng  Mat
Kims     1    87    80
Lees     1    79    90
Parks    2    80    80
```

[] 데이터 프레임 데이터 추출

> 열 변수를 사용한 열의 행 추출

```
In [35]: ## [Data Frame] 데이터 추출 : 열 변수로 열 추출
df['Eng']
```

```
Out[35]: Kims    87
Lees    79
Parks   80
Name: Eng, dtype: int64
```

```
In [36]: ## [Data Frame] 데이터 추출 : 열 변수로 열 추출
df[['Eng', 'Mat']]
```

```
Out[36]:   Eng  Mat
Kims    87    80
Lees    79    90
Parks   80    80
```

```
In [37]: ## [Data Frame] 데이터 추출 : 열 변수로 열 추출
x = df['Eng'];           # 'Eng' 열 값을 시리즈(Series)로 추출
```

```
print(type(x))
x

<class 'pandas.core.series.Series'>
Kims      87
Lees      79
Parks     80
Name: Eng, dtype: int64
```

```
In [47]: ## [Data Frame] 데이터 추출 : 열 변수로 열 추출
x = df[['Eng']]           # 'Eng' 열 값들을 시리즈(Series)로 추출
print(type(x))
x
```

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

```
Out[47]:      Eng
Kims    87
Lees    79
Parks   80
```

```
In [38]: ## [Data Frame] 데이터 추출 : 열 변수로 열 추출
x = df[['Eng', 'Mat']];      # 'Eng'와 'Mat' 열 값들을 Data Frame으로 추출
print(type(x))
x
```

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

```
Out[38]:      Eng  Mat
Kims    87    80
Lees    79    90
Parks   80    80
```

```
In [39]: ## [Data Frame] 데이터 추출 : 원하는 열의 행 추출
collist = ['Eng', 'Mat']
x = df[collist];           # 'Eng'와 'Mat'는 List 형태로 입력
x
```

```
Out[39]:      Eng  Mat
Kims    87    80
Lees    79    90
Parks   80    80
```

> 열을 추출하여 조건식을 사용한 행 추출

```
In [41]: ## [Data Frame] 데이터 추출 : 조건부 행 검사
df['Class'] == 1
```

```
Out[41]: Kims      True
Lees      True
Parks    False
Name: Class, dtype: bool
```

```
In [42]: ## [Data Frame] 데이터 추출 : 조건부 행 추출
df[df['Class'] == 1] # df에서 (df['Class'] == 1)가 True인 것만 추출
```

Out[42]:

	Class	Eng	Mat
Kims	1	87	80
Lees	1	79	90

In [43]: ## [Data Frame] 데이터 추출 : 조건부 행 추출
x = df[df['Eng'] >= 80]; # 'Eng' 열 값들을 시리즈(Series)로 추출해서 비교
x

Out[43]:

	Class	Eng	Mat
Kims	1	87	80
Parks	2	80	80

In [44]: ## [Data Frame] 데이터 추출 : 조건부 행 추출
x = df[(df['Eng'] >= 80) & (df['Mat'] >= 80)];
x

Out[44]:

	Class	Eng	Mat
Kims	1	87	80
Parks	2	80	80

> 행 추출하여 열 선택 추출

In [53]: ## [Data Frame] 데이터 추출: 조건부 행, 열 추출
df[df['Class'] == 1]['Mat']

Out[53]: Kims ... 80
Lees ... 90
Name: Mat, dtype: int64

In [55]: ## [Data Frame] 데이터 추출: 조건부 행, 열 추출
df[df['Class'] == 1][['Mat', 'Eng']]

Out[55]:

	Mat	Eng
Kims	80	87
Lees	90	79

In [46]: ## [Data Frame] 데이터 추출: 조건부 행, 열 추출
x = df[(df['Eng'] >= 80) & (df['Mat'] >= 80)][['Eng', 'Mat']]; ### 'Eng', 'Mat' 열
x

Out[46]:

	Eng	Mat
Kims	87	80
Parks	80	80

(알아 두면 좋아요) .을 이용해 변수를 간단하게 추출하기

> DataFrame.Col_variable로 열 추출 가능

```
In [51]: ## [Data Frame] 데이터 추출 :  
df.Mat
```

```
Out[51]: Kims     80  
Lees      90  
Parks     80  
Name: Mat, dtype: int64
```

```
In [ ]:
```

[] 데이터 추출 : loc[] 이용

> Index는 행을 식별하는 Key 역할을 함.

> Indexing은 Index를 이용해 데이터를 추출하는 방법 > 결과적으로 행을 추출

> loc[]는 데이터프레임에서 특정 행과 열을 선택하는 인덱싱 방법 중 하나

> loc[row_index, col_index]로 추출

> loc[x:y] 숫자 index가 x 이상 ~ y 이하 행을 추출 >> iloc[x:y] x 이상 ~ y 미만 행을 추출

```
In [153...]: ## [Data Frame] 생성  
import pandas as pd  
df = pd.DataFrame({'Class' : [1, 1, 2],  
                   'Eng' : [87, 79, 80],  
                   'Mat' : [80, 90, 80]},  
                   index = ['Kims', 'Lees', 'Parks']) #index 강제 부여  
df
```

```
Out[153]:
```

	Class	Eng	Mat
Kims	1	87	80
Lees	1	79	90
Parks	2	80	80

```
In [154...]: ## [Data Frame] 데이터 추출: loc[] > 행 (Series 자료형) 추출  
# loc[row_index, col_index]  
result = df.loc['Kims'] #결과는 Series 자료형  
print(type(result))  
result
```

```
<class 'pandas.core.series.Series'>  
Out[154]:
```

Class	1
Eng	87
Mat	80
Name:	Kims, dtype: int64

```
In [78]: ## [Data Frame] 데이터 추출: loc[] > 행 (Data Frame 자료형) 추출  
result = df.loc[['Kims']] #결과는 Data Frame 자료형  
print(type(result))  
result
```

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

Out[78]:

	Class	Eng	Mat
Kims	1	87	80

In [79]:

```
## [Data Frame] 데이터 추출: loc[] > 복수 행 추출
result = df.loc[:] #전체 행 선택, 결과는 Data Frame 자료형
result = df.loc[['Kims', 'Lees']] #결과는 Data Frame 자료형
result
```

Out[79]:

	Class	Eng	Mat
Kims	1	87	80
Lees	1	79	90

In [80]:

```
## [Data Frame] 데이터 추출: loc[] > 행 (Data Frame 자료형) 추출
result = df.loc['Kims':'Lees'] #결과는 Data Frame 자료형
print(type(result))
result
```

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

Out[80]:

	Class	Eng	Mat
Kims	1	87	80
Lees	1	79	90

> 행 추출하여 열 선택 추출

In [81]:

```
## [Data Frame] 데이터 추출: loc[] > 복수 행, 선택 열 추출
# loc[row_index, col_index]
result = df.loc[['Kims', 'Lees'], 'Mat'] #결과는 Series 자료형
result = df.loc[['Kims', 'Lees'], ['Mat']] #결과는 Data Frame 자료형
result
```

Out[81]:

	Mat
Kims	80
Lees	90

In [82]:

```
## [Data Frame] 데이터 추출: loc[] > 복수 행, 선택 열 추출
result = df.loc[['Kims', 'Lees'], :] #전체 행 선택, #결과는 Data Frame 자료형
result = df.loc[['Kims', 'Lees'], ['Mat', 'Eng']] #결과는 Data Frame 자료형
result
```

Out[82]:

	Mat	Eng
Kims	80	87
Lees	90	79

In [83]:

```
## [Data Frame] 데이터 추출: loc[] > 조건부 추출
result = df.loc[df['Eng'] >= 80]
result
```

Out[83]:

	Class	Eng	Mat
Kims	1	87	80
Parks	2	80	80

> Index를 이용한 추출

In [86]:

```
## [Data Frame] 생성
import pandas as pd
df = pd.DataFrame({ 'Name' : ['Kims', 'Lees', 'Parks'],
                     'Class' : [1, 1, 2],
                     'Eng' : [87, 79, 80],
                     'Mat' : [80, 90, 80]}) #index 자동 부여
df
```

Out[86]:

	Name	Class	Eng	Mat
0	Kims	1	87	80
1	Lees	1	79	90
2	Parks	2	80	80

In [89]:

```
## [Data Frame] 데이터 추출: index로 추출
df.loc[0] #0번 index 행 추출
```

Out[89]:

```
Name      Kims
Class      1
Eng       87
Mat       80
Name: 0, dtype: object
```

In [92]:

```
## [Data Frame] 데이터 추출: index로 추출
df.loc[[0, 2]] #0, 2번 index 행 추출
```

Out[92]:

	Name	Class	Eng	Mat
0	Kims	1	87	80
2	Parks	2	80	80

In [93]:

```
## [Data Frame] 데이터 추출: index로 추출
df.loc[0:2] #0~2번 index 행 추출
```

Out[93]:

	Name	Class	Eng	Mat
0	Kims	1	87	80
1	Lees	1	79	90
2	Parks	2	80	80

In [94]:

```
## [Data Frame] 데이터 추출: index로 추출
df.loc[1:] #1에서 끝까지 index 행 추출
```

```
Out[94]:
```

	Name	Class	Eng	Mat
1	Lees	1	79	90
2	Parks	2	80	80

> 행 추출하여 열 선택 추출

```
In [95]: ## [Data Frame] 데이터 추출: index로 추출  
df.loc[:1, ['Eng', 'Mat']] #1에서 끝까지 index 행 추출
```

```
Out[95]:
```

	Eng	Mat
0	87	80
1	79	90

```
In [ ]:
```

> index 번호가 없을 때, index 번호로 .loc[:] 사용 불가능

```
In [ ]: ## [Data Frame] 데이터 추출:  
## index 번호가 없을 때, index 번호로 .loc[:] 사용 불가능  
df.loc[0:1]
```

> index 번호가 없을 때, index 번호로 .loc[:] 사용 가능

```
In [107...]: ## [Data Frame] 데이터 추출:  
## index 번호가 없을 때, index 번호로 .iloc[:] 사용 가능  
df.iloc[0:1]
```

```
Out[107]:
```

	Class	Eng	Mat
Kims	1	87	80

```
In [ ]:
```

[] 데이터 추출 : iloc[] 이용

> iloc[]는 데이터 추출 전에 행과 열에 임시로 숫자 index를 지정하여 사용하는 인덱싱 방법

> loc[]와 달리 조전부 추출이 불가능

> iloc[x:y] x 이상 ~ y 미만 행을 추출 >> loc[x:y] 숫자 index가 x 이상 ~ y 이하 행을 추출

```
In [123...]:
```

```
## [Data Frame] 생성  
import pandas as pd  
df = pd.DataFrame({ 'Name' : ['Kims', 'Lees', 'Parks'],  
                    'Class' : [1, 1, 2],  
                    'Eng' : [87, 79, 80],  
                    'Mat' : [80, 90, 80]}) #index 자동 부여  
df
```

```
Out[123]:   Name  Class  Eng  Mat
```

0	Kims	1	87	80
1	Lees	1	79	90
2	Parks	2	80	80

```
In [124...]
```

```
## [Data Frame] 데이터 추출: iloc[] > 행 추출
result = df.iloc[0]          #1개 데이터이므로 Series 형으로 추출
result = df.iloc[[0]]        #Data Frame 형으로 추출
result = df.iloc[:]          #복수개 데이터이므로 Data Frame 형으로 추출
result = df.iloc[1:3]         #복수개 데이터이므로 Data Frame 형으로 추출
result
```

```
Out[124]:
```

	Name	Class	Eng	Mat
1	Lees	1	79	90
2	Parks	2	80	80

```
In [125...]
```

```
## [Data Frame] 데이터 추출: iloc[] > 행 추출
result = df.iloc[[0, 2]]      #복수개 데이터이므로 Data Frame 형으로 추출
result
```

```
Out[125]:
```

	Name	Class	Eng	Mat
0	Kims	1	87	80
2	Parks	2	80	80

Index 번호로 열 추출

```
In [126...]
```

```
## [Data Frame] 데이터 추출: iloc[] > 행, 열 추출
result = df.iloc[[0, 2], [1]]      #열 선택도 열 순서 번호(index)로 명시
result = df.iloc[[0, 2], [1, 0]]    #열 선택도 열 순서 번호(index)로 명시
result
```

```
Out[126]:
```

	Class	Name
0	1	Kims
2	2	Parks

> iloc[]는 조건 추출이 불가능

```
In [131...]
```

```
df.iloc[:, [1]]
```

```
Out[131]:
```

	Class
0	1
1	1
2	2

```
In [135...]
```

```
df.iloc[df.iloc[:, [1]] == 1]
```

```

-----
--> IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_28148\3523535584.py in <module>
----> 1 df.iloc[df.iloc[:, [1]] == 1]

~\anaconda3\lib\site-packages\pandas\core\indexing.py in __getitem__(self, key)
    965
    966     .... maybe_callable = com.apply_if_callable(key, self.obj)
--> 967         return self._getitem_axis(maybe_callable, axis=axis)
    968
    969     ... def _is_scalar_access(self, key: tuple):

~\anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_axis(self, key, axis)
    1492             key = slice(None)
    1493         elif isinstance(key, ABCDataFrame):
--> 1494             raise IndexError(
    1495                 "DataFrame indexer is not allowed for .iloc\n"
    1496                 "Consider using .loc for automatic alignment."

```

IndexError: DataFrame indexer is not allowed for .iloc
Consider using .loc for automatic alignment.

<> loc[x:y]가 x 이상 ~ y 이하 행을 추출하는 이유

> index가 사용자가 직접 부여하므로 0부터 시작한다는 보장이 없음

> 따라서 index 범위는 콜 찍어서 loc[부터:까지]가 된다.

```
In [114]: ## [Data Frame] 생성
import pandas as pd
df = pd.DataFrame({'Class' : [1, 1, 2],
                   'Eng' : [87, 79, 80],
                   'Mat' : [80, 90, 80]},
                  index = [1, 2, 3]) #index 강제 부여
df
```

```
Out[114]:   Class  Eng  Mat
      1      1    87    80
      2      1    79    90
      3      2    80    80
```

```
In [115]: df.loc[1:2]
```

```
Out[115]:   Class  Eng  Mat
      1      1    87    80
      2      1    79    90
```

<> iloc[x:y]가 x 이상 ~ y 미만 행을 추출하는 이유

> index가 0부터 자동 부여되므로 0부터 시작한다는 보장이 있음

> 따라서 index 범위는 range() 처럼 iloc[부터:전까지]가 된다.

```
In [ ]: ## [Data Frame] 생성
import pandas as pd
df = pd.DataFrame({ 'Name' : ['Kims', 'Lees', 'Parks'],
                     'Class' : [1, 1, 2],
                     'Eng' : [87, 79, 80],
                     'Mat' : [80, 90, 80]}) #index 자동 부여
df
```

```
In [116... df.iloc[1:2]
```

```
Out[116]: Class Eng Mat
          2    1   79   90
```

```
In [ ]:
```

(알아 두면 좋아요) 열 변수를 index로 전환하기

> index를 열 변수로 전환도 가능(단, index에 변수명이 있을 때만 가능)

```
In [136... ## [Data Frame] 생성
import pandas as pd
df = pd.DataFrame({ 'Name' : ['Kims', 'Lees', 'Parks'],
                     'Class' : [1, 1, 2],
                     'Eng' : [87, 79, 80],
                     'Mat' : [80, 90, 80]},) #index 강제 부여
df
```

```
Out[136]: Name Class Eng Mat
          0   Kims    1   87   80
          1   Lees    1   79   90
          2   Parks   2   80   80
```

```
In [119... # 'Name' 열을 인덱스로 설정
df.set_index('Name', inplace=True)
df
```

```
Out[119]: Class Eng Mat
          Name
          Kims    1   87   80
          Lees    1   79   90
          Parks   2   80   80
```

```
In [120... # 현재 인덱스를 일반 열로 변경
df.reset_index(inplace=True)
df
```

Out[120]:

	Name	Class	Eng	Mat
0	Kims	1	87	80
1	Lees	1	79	90
2	Parks	2	80	80

In []:

[] 데이터 프레임 데이터 조작

In [159...]

```
## [Data Frame] 생성
import pandas as pd
df = pd.DataFrame({ 'Name' : ['Kims', 'Lees', 'Parks'],
                     'Class' : [1, 1, 2],
                     'Eng' : [87, 79, 80],
                     'Mat' : [80, 90, 80]},) #index 강제 부여
df
```

Out[159]:

	Name	Class	Eng	Mat
0	Kims	1	87	80
1	Lees	1	79	90
2	Parks	2	80	80

In [160...]

```
## [Data Frame] 조작 : 열 추가
df['avg'] = (df['Eng'] + df['Mat']) / 2 # 'avg'열을 만들어서 평균 값 추가
df
```

Out[160]:

	Name	Class	Eng	Mat	avg
0	Kims	1	87	80	83.5
1	Lees	1	79	90	84.5
2	Parks	2	80	80	80.0

In [161...]

```
## [Data Frame] 조작 : 열 순서 변경
df = df[['Class', 'Name', 'Eng', 'Mat', 'avg']] # 원하는 컬럼 순으로 재정렬
df
```

Out[161]:

	Class	Name	Eng	Mat	avg
0	1	Kims	87	80	83.5
1	1	Lees	79	90	84.5
2	2	Parks	80	80	80.0

In [162...]

```
## [Data Frame] 조작 : 열 순서 변경
col_seq = ['Class', 'Name', 'Eng', 'Mat', 'avg']
df = df[col_seq] # 원하는 컬럼 순으로 재정렬
df
```

Out[162]:

	Class	Name	Eng	Mat	avg
0	1	Kims	87	80	83.5
1	1	Lees	79	90	84.5
2	2	Parks	80	80	80.0

[] 데이터프레임 개별 항목 값 조작

> 반복문을 사용하여 조작

In [163...]

```
## [Data Frame] 조작 : 반복문 사용 접근
for index, row in df.iterrows():
    print(index, df.at[index, 'Mat']) # .at[]은 행과 열로 특정값 검색
```

0 80
1 90
2 80

In [164...]

```
## [Data Frame] 조작 : 반복문 사용 조작
for index, row in df.iterrows():
    if df.at[index, 'Mat'] <= 90:
        df.at[index, 'Mat'] = df.at[index, 'Mat'] + 10
df
```

Out[164]:

	Class	Name	Eng	Mat	avg
0	1	Kims	87	90	83.5
1	1	Lees	79	100	84.5
2	2	Parks	80	90	80.0

In [165...]

```
## [Data Frame] 조작 : 열 값 일괄 적용 조작
df['Mat'] = df['Mat'] - 10
df
```

Out[165]:

	Class	Name	Eng	Mat	avg
0	1	Kims	87	80	83.5
1	1	Lees	79	90	84.5
2	2	Parks	80	80	80.0

In []:

[] 데이터 프레임 관련 함수 사용

In [167...]

```
## [Data Frame] 관련 함수 : 통계 보기
sum(df['Eng'])
```

Out[167]:

246

In [168...]

```
## [Data Frame] 관련 함수 : 통계 보기
len(df)
```

Out[168]: 3

```
In [169... ## [Data Frame] 관련 함수 : 통계 보기  
avg_mat = sum(df['Mat']) / len(df)  
print('Average of Math : ', avg_mat)
```

Average of Math : 83.33333333333333

```
In [170... ## [Data Frame] 관련 함수 사용  
x = df['Mat']; # 'Mat' 열 값들을 시리즈로 추출  
x.mean() # 'Mat' 열 값들의 평균 구하기
```

Out[170]: 83.33333333333333

```
In [171... ## [Data Frame] 관련 함수 사용  
x = df['Mat']; # 'Mat' 열 값들을 시리즈로 추출  
x.value_counts() # 'Mat' 열 값들의 개수 구하기
```

Out[171]:
80 2
90 1
Name: Mat, dtype: int64

데이터 정렬

```
In [172... ## [Data Frame] 관련 함수 : sorting된 Data Frame 생성  
sdf = df.sort_values(by = 'avg') # 'avg' 컬럼 값 순으로 정렬  
sdf
```

Out[172]:

	Class	Name	Eng	Mat	avg
2	2	Parks	80	80	80.0
0	1	Kims	87	80	83.5
1	1	Lees	79	90	84.5

```
In [177... ## [Data Frame] 관련 함수 : sorting된 Data Frame 생성  
sdf = df.sort_values(by = 'avg', ascending=False)  
sdf
```

Out[177]:

	Class	Name	Eng	Math	avg
1	1	Lees	79	90	84.5
0	1	Kims	87	80	83.5
2	2	Parks	80	80	80.0

컬럼명 변경

```
In [178... ## [Data Frame] 관련 함수 : 컬럼명 변경  
df = df.rename(columns = {'Mat' : 'Math'}) # 컬럼명 변경  
df
```

Out[178]:

	Class	Name	Eng	Math	avg
0	1	Kims	87	80	83.5
1	1	Lees	79	90	84.5
2	2	Parks	80	80	80.0

행, 열 제거

In [179...]

```
## [Data Frame] 관련 함수 : 행, 열 제거
ddf = df.drop( 'avg', axis = 'columns' )           # 컬럼 이름으로 컬럼 제거
```

Out[179]:

	Class	Name	Eng	Math
0	1	Kims	87	80
1	1	Lees	79	90
2	2	Parks	80	80

In [180...]

```
## [Data Frame] 관련 함수 : 행, 열 제거
ddf = df.drop( 1, axis = 'rows' )                  # 행 이름으로 열 제거
ddf
```

Out[180]:

	Class	Name	Eng	Math	avg
0	1	Kims	87	80	83.5
2	2	Parks	80	80	80.0

행 일부만 출력

In [182...]

```
## [Data Frame] 관련 함수 : 일부만 출력
df.head(2)                                         # 데이터 행 상위 일부만 출력 (숫자 생략하면 알아서)
```

Out[182]:

	Class	Name	Eng	Math	avg
0	1	Kims	87	80	83.5
1	1	Lees	79	90	84.5

In [181...]

```
## [Data Frame] 관련 함수 : 일부만 출력
df.tail(1)                                         # 데이터 행 하위 일부만 출력 (숫자 생략하면 알아서)
```

Out[181]:

	Class	Name	Eng	Math	avg
2	2	Parks	80	80	80.0

행과 열의 수 출력

In [183...]

```
## [Data Frame] 관련 함수 : 행, 열의 수 출력
df.shape
```

Out[183]:

(3, 5)

프레임 속성 출력

```
In [184...]: ## [Data Frame] 관련 함수 : 데이터 프레임 속성 출력  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 3 entries, 0 to 2  
Data columns (total 5 columns):  
 # ... Column Non-Null Count Dtype ...  
 --- ... ---  
 0 Class    3 non-null    int64  
 1 Name     3 non-null    object  
 2 Eng      3 non-null    int64  
 3 Math     3 non-null    int64  
 4 avg      3 non-null    float64  
dtypes: float64(1), int64(3), object(1)  
memory usage: 248.0+ bytes
```

프레임 통계 출력

```
In [185...]: ## [Data Frame] 관련 함수 : 통계 보기  
df.describe()
```

```
Out[185]:
```

	Class	Eng	Math	avg
count	3.000000	3.000000	3.000000	3.000000
mean	1.333333	82.000000	83.333333	82.666667
std	0.577350	4.358899	5.773503	2.362908
min	1.000000	79.000000	80.000000	80.000000
25%	1.000000	79.500000	80.000000	81.750000
50%	1.000000	80.000000	80.000000	83.500000
75%	1.500000	83.500000	85.000000	84.000000
max	2.000000	87.000000	90.000000	84.500000

```
In [ ]:
```

□ Excel이나 csv 타입 파일로부터 데이터 프레임 구성하기

> import pandas 를 통해 read_excel() 또는 read_csv()를 사용한다.

>> 데이터에 한글이나 특수 문자가 포함되어 있는 경우 encoding='UTF-8' 적용

```
In [186...]: ## [Data Frame] Excel 파일 >> 데이터 프레임 구축  
import pandas as pd  
df_exam = pd.read_excel('excel_exam.xlsx', header=None) #header가 없을 때, 데이터  
df_exam = pd.read_excel('excel_exam.xlsx') #header가 있을 때, 첫 행이  
df_exam
```

Out[186]:

	id	nclass	math	english	science
0	1	1	50	98	50
1	2	1	60	97	60
2	3	1	45	86	78
3	4	1	30	98	58
4	5	2	25	80	65
5	6	2	50	89	98
6	7	2	80	90	45
7	8	2	90	78	25
8	9	3	20	98	15
9	10	3	50	98	45
10	11	3	65	65	65
11	12	3	45	85	32
12	13	4	46	98	65
13	14	4	48	87	12
14	15	4	75	56	78
15	16	4	58	98	65
16	17	5	65	68	98
17	18	5	80	78	90
18	19	5	89	68	87
19	20	5	78	83	58

In [187...]

```
## [Data Frame] Excel 파일 >> 데이터 프레임 구축
import pandas as pd
df_exam = pd.read_excel('excel_exam.xlsx', sheet_name='Sheet2', header=None)    #시트 이름
#header가 없을 때, 데이터 프레임에 열 index가 자동 부여됨
df_exam
```

Out[187]:

	0	1	2	3	4
0	1	1	50	98	50
1	2	1	60	97	60
2	3	1	45	86	78
3	4	1	30	98	58
4	5	2	25	80	65
5	6	2	50	89	98
6	7	2	80	90	45
7	8	2	90	78	25
8	9	3	20	98	15
9	10	3	50	98	45
10	11	3	65	65	65
11	12	3	45	85	32
12	13	4	46	98	65
13	14	4	48	87	12
14	15	4	75	56	78
15	16	4	58	98	65
16	17	5	65	68	98
17	18	5	80	78	90
18	19	5	89	68	87
19	20	5	78	83	58

In [188...]

```
## [Data Frame] 데이터 프레임 >> csv 파일로 저장
import pandas as pd
df_exam = pd.read_excel('excel_exam.xlsx')      #header가 있을 때, 첫 행이 데이터 프레임의 헤더로 사용되는 경우
df_exam.to_csv("exam_csv.csv")
df_exam.to_csv("exam_csv.csv", index=False) #index는 빼고 저장

df_exam = pd.read_csv('exam_csv.csv')      #header가 있을 때, 첫 행이 데이터 프레임의 헤더로 사용되는 경우
```

Out[188]:

	id	nclass	math	english	science
0	1	1	50	98	50
1	2	1	60	97	60
2	3	1	45	86	78
3	4	1	30	98	58
4	5	2	25	80	65
5	6	2	50	89	98
6	7	2	80	90	45
7	8	2	90	78	25
8	9	3	20	98	15
9	10	3	50	98	45
10	11	3	65	65	65
11	12	3	45	85	32
12	13	4	46	98	65
13	14	4	48	87	12
14	15	4	75	56	78
15	16	4	58	98	65
16	17	5	65	68	98
17	18	5	80	78	90
18	19	5	89	68	87
19	20	5	78	83	58

In [189...]

```
def int2han(remain):
    how = [1000, 100, 10, 1]
    how_han = ['천', '백', '십', '']
    digit_han = ['', '일', '이', '삼', '사', '오', '육', '칠', '팔', '구']
    result = ''
    for i in range(len(how)):
        out = remain // how[i]
        if out != 0:
            result += digit_han[out] + how_han[i]
        remain = remain % how[i]
    return result

inn = 123
result = int2han(inn)
print(result)
```

일백이십삼

In []:

[실습-1] 데이터 프레임 조작하기

1. Excel > Data Frame 변환

> 'excel_exam.xlsx' 파일을 읽어서 데이터 프레임 구성

2. 'total'과 'avg' 열 변수 추가

> 'total'은 'math', 'english','science'의 합

> 'avg'는 'math', 'english','science'의 평균

3. 'pass' 열 변수를 추가

> 'pass'를 'None' 값으로 초기화

4. 'pass' 값을 변경

> 'pass'는 'avg' 값이 60 이상이면 'P', 아니면 'F'로 값 변경

5. 열과 행이 선택된 'df_pass' 데이터 프레임 생성

> 'pass'가 'P' 행들만 선택

> 선택된 행 중에 'id', 'nclass', 'math', 'english', 'science', 'avg' 열만 추출해서 구성

> 열 순서를 'nclass', 'id', 'math', 'english', 'science', 'avg' 순으로 구성

6. Data Frame > csv 파일로 변환

> 파일로 변환하기 전에 'df_pass' 데이터 프레임을 'avg' 내림차 순으로 정렬

> 데이터 프레임을 'excel_exam_pass.csv' 파일로 저장 (단, index는 빼고)

7. Data Frame > csv 파일로 변환

> 'excel_exam_pass.csv' 파일을 데이터 프레임으로 읽어서, 출력 확인

In []:

정리하기

In [3]: import pandas as pd

```
# 1. 데이터 프레임 만들기
df = pd.DataFrame({'name' : ['김지훈', '이유진', '박동현', '김민지'],
                    'english' : [90, 80, 60, 70],
                    'math' : [50, 60, 100, 20]})
```

```
df_midterm = pd.DataFrame({'english' : [90, 80, 60, 70],
                            'math' : [50, 60, 100, 20],
                            'nclass' : [1, 1, 2, 2]})
```

```
# 2. 외부 데이터 이용하기
```

```
# 엑셀 파일 불러오기
```

```
df_exam = pd.read_excel('excel_exam.xlsx')

# CSV 파일 불러오기
df_csv_exam = pd.read_csv('exam.csv')

# CSV 파일로 저장하기
df_midterm.to_csv('output_newdata.csv')
```

In []: