
▣ Ch04 데이터 프레임(Data Frame) 자료 다루기

- > 외장 Pandas 자료형
- > pandas라는 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 index로 식별된다.
- > 데이터 프레임의 데이터를 구성하는 열 값을 시리즈로 추출하여 조작할 수 있다.
- > pandas 패키지를 통해 사용할 수 있다.

```
In [35]: ## 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[35]: 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 [65]: ## [Data Frame] 생성
import pandas as pd
df = pd.DataFrame({'Eng' : [87, 79, 80],
                   'Mat' : [80, 90, 80]}) #index 자동 부여
df
```

```
Out[65]:   Eng  Mat
0    87    80
1    79    90
2    80    80
```

```
In [160... ## [Data Frame] 생성
import pandas as pd
df = pd.DataFrame({'Eng' : [87, 79, 80],
                   'Mat' : [80, 90, 80]},
                   index = ['Kims', 'Lees', 'Parks']) #index 강제 부여
df
```

```
Out[160]:   Eng  Mat
Kims    87    80
Lees     79    90
Parks    80    80
```

<> 데이터 프레임 데이터 추출

```
In [35]: ## [Data Frame] 데이터 추출 : 원하는 열의 행 추출
x = df['Eng'];           # 'Eng' 열 값들을 시리즈(Series)로 추출
type(x)
x
```

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

```
In [34]: ## [Data Frame] 데이터 추출 : 원하는 열의 행 추출
x = df[['Eng', 'Mat']]; # 'Eng'와 'Mat' 열 값들을 Data Frame으로 추출
type(x)
x
```

```
Out[34]:   Eng  Mat
Kims    87    80
Lees     79    90
Parks    80    80
```

```
In [36]: ## [Data Frame] 데이터 추출 : 원하는 열의 행 추출
collist = ['Eng', 'Mat']
```

```
x = df[collist]; # 'Eng'와 'Mat'는 List 형태로 입력  
x
```

Out[36]:

	Eng	Mat
Kims	87	80
Lees	79	90
Parks	80	80

In [37]:

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

Out[37]:

	Eng	Mat
Kims	87	80
Parks	80	80

In [38]:

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

Out[38]:

	Eng	Mat
Kims	87	80
Parks	80	80

<> 데이터 추출 : loc[] 이용

> **loc[]**는 데이터프레임에서 특정 행과 열을 선택하거나 조작하기 위한 인덱싱 방법 중 하나

> **loc[x:y]** 숫자 **index**가 x 이상 ~ y 이하 행을 추출 >> **iloc[x:y]** x 이상 ~ y 미만 행을 추출

In [27]:

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

```
<class 'pandas.core.series.Series'>
```

Out[27]:

```
Eng ... 87  
Mat ... 80  
Name: Kims, dtype: int64
```

In [28]:

```
## [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[28]:

	Eng	Mat
Kims	87	80

In [42]:

```
## [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[42]:

	Eng	Mat
Kims	87	80
Lees	79	90

In [30]:

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

Out[30]:

	Mat
Kims	80
Lees	90

In [39]:

```
## [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[39]:

	Mat	Eng
Kims	80	87
Lees	90	79

In [53]:

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

Out[53]:

	Eng	Mat
Kims	87	80
Parks	80	80

<> 데이터 추출 : iloc[] 이용

> **iloc[]**는 추출 전에 행과 열에 숫자 **index**를 지정하여 특정 행과 열을 선택하거나 조작하기 위한 인덱싱 방법 중 하나

> **loc[]**와 달리 조건부 추출이 불가능

> **iloc[x:y]** x 이상 ~ y 미만 행을 추출 >> **loc[x:y]** 숫자 **index**가 x 이상 ~ y 이하 행을 추출

```
In [77]: ## [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[77]: **Eng Mat**

	Eng	Mat
1	79	90
2	80	80

```
In [73]: ## [Data Frame] 데이터 추출: iloc[] > 행 추출
result = df.iloc[[0, 2]]      #복수개 데이터이므로 Data Frame 형으로 추출
result
```

Out[73]: **Eng Mat**

	Eng	Mat
0	87	80
2	80	80

```
In [76]: ## [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[76]: **Mat Eng**

	Mat	Eng
0	80	87
2	80	80

<> 데이터 프레이 데이터 조작

```
In [161...]: ## [Data Frame] 조작 : 열 추가
df['avg'] = (df['Eng'] + df['Mat']) / 2           # 'avg' 열을 만들어서 평균 값 추가
df
```

Out[161]: **Eng Mat avg**

	Eng	Mat	avg
Kims	87	80	83.5
Lees	79	90	84.5
Parks	80	80	80.0

```
In [162...]: ## [Data Frame] 조작 : 열 순서 변경
df = df[['Mat', 'Eng', 'avg']]           # 원하는 컬럼 순으로 재정렬
df

col_seq = ['Mat', 'Eng', 'avg']
df = df[col_seq]                         # 원하는 컬럼 순으로 재정렬
df
```

Out[162]:

	Mat	Eng	avg
Kims	80	87	83.5
Lees	90	79	84.5
Parks	80	80	80.0

<> 데이터프레임 개별 항목 값 조작

In [164...]

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

Kims 80
Lees 90
Parks 80

In [165...]

```
## [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[165]:

	Mat	Eng	avg
Kims	90	87	83.5
Lees	100	79	84.5
Parks	90	80	80.0

In [169...]

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

Out[169]:

	Mat	Eng	avg
Kims	70	87	83.5
Lees	80	79	84.5
Parks	70	80	80.0

<> 데이터 프레임 관련 함수 사용

In [105...]

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

Average of Math : 83.33333333333333

In [106...]

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

```
Out[106]: 80    2  
90    1  
Name: Mat, dtype: int64
```

데이터 정렬

```
In [107...]: ## [Data Frame] 관련 함수 : sorting된 Data Frame 생성  
sdf = df.sort_values(by = 'avg')          # 'avg' 컬럼 값 순으로 정렬  
sdf  
sdf = df.sort_values(by = 'avg', ascending=False)  
sdf
```

```
Out[107]:   Mat  Eng  avg  
  
Lees    90   79  84.5  
  
Kims    80   87  83.5  
  
Parks   80   80  80.0
```

컬럼명 변경

```
In [108...]: ## [Data Frame] 관련 함수 : 컬럼명 변경  
df = df.rename(columns = {'Mat' : 'Math'})           # 컬럼명 변경  
df
```

```
Out[108]:   Math  Eng  avg  
  
Kims    80   87  83.5  
  
Lees    90   79  84.5  
  
Parks   80   80  80.0
```

행, 열 제거

```
In [109...]: ## [Data Frame] 관련 함수 : 행, 열 제거  
ddf = df.drop( 'Lees', axis = 'rows')           # 행 이름으로 열 제거  
ddf = ddf.drop( 'avg', axis = 'columns')        # 컬럼 이름으로 컬럼 제거  
ddf
```

```
Out[109]:   Math  Eng  
  
Kims    80   87  
  
Parks   80   80
```

```
In [110...]: ## [Data Frame] 관련 함수 : 일부만 출력  
df.head(2)           # 데이터 행 상위 일부만 출력 (숫자 생략하면 알아서)  
df.tail(1)           # 데이터 행 하위 일부만 출력 (숫자 생략하면 알아서)
```

```
Out[110]:   Math  Eng  avg  
  
Parks   80   80  80.0
```

```
In [174...]: ## [Data Frame] 관련 함수 : 행, 열의 수 출력  
df.shape
```

```
Out[174]: (3, 3)
```

```
In [175...]: ## [Data Frame] 관련 함수 : 데이터 프레임 속성 출력  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Index: 3 entries, Kims to Parks  
Data columns (total 3 columns):  
 # ... Column Non-Null Count Dtype ...  
 --- ...  
 0 Mat    3 non-null    int64  
 1 Eng    3 non-null    int64  
 2 avg    3 non-null    float64  
dtypes: float64(1), int64(2)  
memory usage: 204.0+ bytes
```

```
In [104...]: ## [Data Frame] 관련 함수 : 통계 보기  
df.describe()
```

```
Out[104]:
```

	Mat	Eng	avg
count	3.000000	3.000000	3.000000
mean	83.333333	82.000000	82.666667
std	5.773503	4.358899	2.362908
min	80.000000	79.000000	80.000000
25%	80.000000	79.500000	81.750000
50%	80.000000	80.000000	83.500000
75%	85.000000	83.500000	84.000000
max	90.000000	87.000000	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 포함되어있는경우 `encoding='UTF-8'` 적용

```
In [148...]: ## [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[148]:

	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 [149...]

```
## [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[149]:

	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 [156...]

```
## [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 사용된다.
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 사용된다.
df_exam
```

Out[156]:

	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 [171...]

```
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 []:

[응용 실습] Excel > Data Frame > csv > Data Frame 변환

> 'excel_exam.xlsx' 파일을 읽어서 데이터 프레임 구성

- > 데이터 프레임의 'id' 값을 int2han()를 사용하여 한글 문자열로 변환
- > 데이터 프레임을 'csv_han.csv' 파일로 저장 (단, index는 빼고)
- > 'csv_han.csv' 파일을 데이터 프레임으로 읽어서, 출력 확인

```
In [170]: ## [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 프레임
for index, row in df_exam.iterrows():
    df_exam.at[index, 'id'] = int2han(df_exam.at[index, 'id'])
df_exam
#df_exam.to_csv("exam_csv.csv", encoding = 'UTF-8') #깨져서 읽힐 경우 한글이나 특수문자
df_exam.to_csv("exam_csv.csv", index=False)

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

Out[170]:

	id	nclass	math	english	science
0	일	1	50	98	50
1	이	1	60	97	60
2	삼	1	45	86	78
3	사	1	30	98	58
4	오	2	25	80	65
5	육	2	50	89	98
6	칠	2	80	90	45
7	팔	2	90	78	25
8	구	3	20	98	15
9	일십	3	50	98	45
10	일십일	3	65	65	65
11	일십이	3	45	85	32
12	일십삼	4	46	98	65
13	일십사	4	48	87	12
14	일십오	4	75	56	78
15	일십육	4	58	98	65
16	일십칠	5	65	68	98
17	일십팔	5	80	78	90
18	일십구	5	89	68	87
19	이십	5	78	83	58

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 [ ]:
```