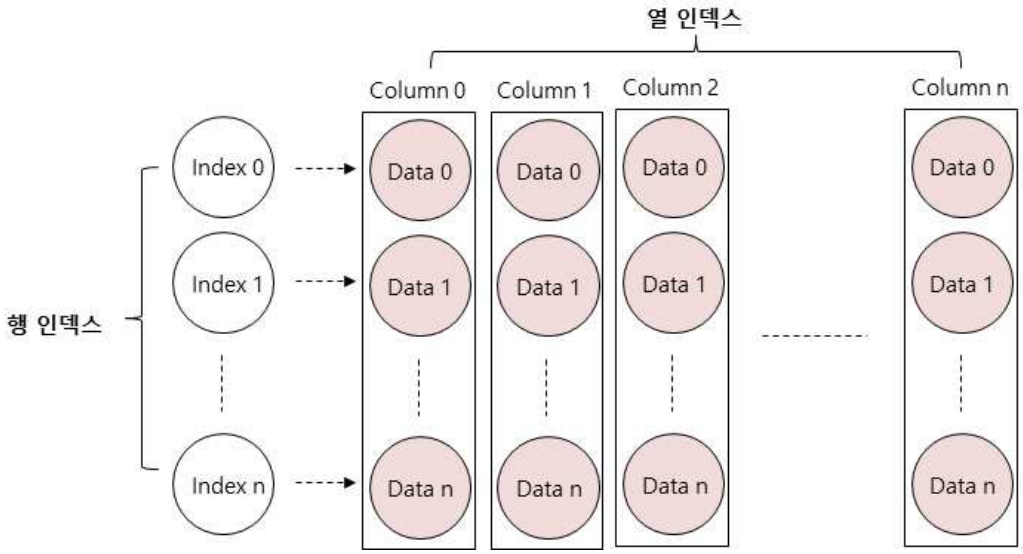
01. 데이터프레임 생성 및 변환

1. Pandas 모듈을 활용한 데이터프레임 생성

○ 데이터프레임이란?

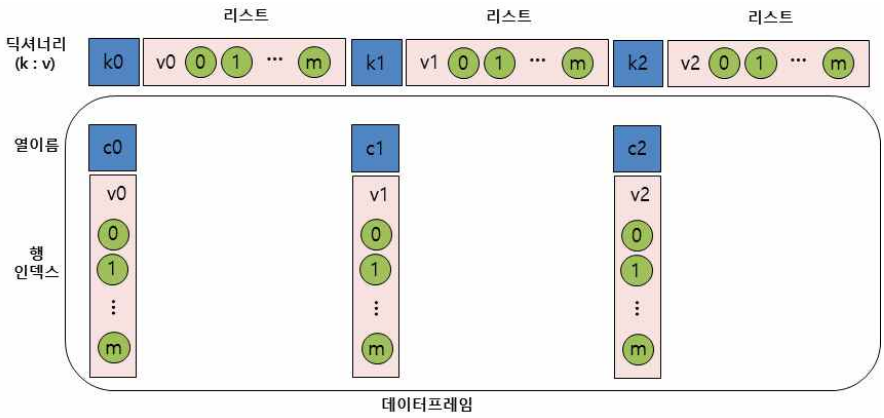
- 행과 열로 만들어진 2차원 배열 구조를 말함.
- 엑셀과 관계형 데이터베이스 등 다양한 분야에서 사용할 수 있음.
- Python의 데이터프레임은 R의 데이터프레임에서 유래되어짐.



- 데이터프레임은 행과 열을 나타내기 위해 두 가지 종류의 주소 (행 인덱스; row index, 열 이름; column name)을 사용함.
- 열은 공통의 속성을 갖는 일련의 데이터를 나타내고, 행은 개별 관측대상에 대한 다양한 속성 데이터들의 모음임. (record)

1) 데이터프레임 만들기

- 여러 개의 시리즈(series)를 모아 놓은 집합을 말합니다. 여기서 시리즈란 1차원 배열을 말합니다. 따라서 데이터프레임을 만들기 위해서는 1차원 배열 여러 개가 필요함.



- 딕셔너리의 값(v)에 해당하는 각 리스트가 시리즈 배열로 변환되어 데이터프레임의 열이 되고, 딕셔너리의 키(k)는 각 시리즈의 이름으로 변환되어 최종적으로 데이터프레임의 열 이름이 됨.

2) 딕셔너리(dictionary) 자료를 데이터프레임(data frame)으로 변환

- 데이터프레임을 만들때는 판다스 모듈의 DataFrame()함수를 사용함. 일반적으로 딕셔너리를 함수의 인자로 전달하는 방식이 주로 활용됨.
- 또한, 데이터프레임의 구조적 특성 때문에 2차원 배열 형태의 데이터를 데이터프레임으로 변환할 수 있음.

pandas.DataFrame(딕셔너리 자료 또는 2차원 배열)

	c0	c1	c2	c3	c4
0	1	4	7	10	13
1	2	5	8	11	14
2	3	6	9	12	15

```
In [1]: import pandas as pd
   ...:
   ...: dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9], 'c3':[10,11,12], 'c4':[13,14,15]}
   ...:
   ...: df = pd.DataFrame(dict_data)

In [2]: print(df)
   c0  c1  c2  c3  c4
0   1   4   7  10  13
1   2   5   8  11  14
2   3   6   9  12  15
```

2. Pandas 모듈을 활용한 데이터프레임 변환

○ 행 인덱스와 열 이름 설정

- 2차원 배열의 경우 딕셔너리 자료와 다르게 행 인덱스와 열 이름이 지정되어 있지 않음. 따라서 2차원 배열을 데이터프레임으로 변환할 때 행 인덱스와 열 이름속성을 직접 지정할 수 있음.
- DataFrame() 함수의 index 옵션을 활용하여 행 인덱스를 지정할 수 있고, columns 옵션을 활용하여 열 이름을 지정할 수 있음.

pandas.DataFrame(2차원 배열,index=행 인덱스 배열,columns=열 이름 배열)

	나이	성별	학교
준서	15	남	덕영중
예은	17	여	수리중

```
In [1]: import pandas as pd
   ...:
   ...: df = pd.DataFrame([[15, '남', '덕영중'], [17, '여', '수리중']],
   ...:                   index=['준서', '예은'],
   ...:                   columns=['나이', '성별', '학교'])

In [2]: print(df)
    나이 성별   학교
준서  15  남  덕영중
예은  17  여  수리중
```

○ 행 인덱스와 열 이름 변경

1) 전체 행 인덱스와 열 이름 변경

- 데이터프레임의 행 인덱스는 index 인덱서를 통해 접근할 수 있고, 열 이름은 columns 인덱서를 통해 접근할 수 있음.
- 이러한 속성을 활용해 데이터프레임 객체의 행 인덱스와 열 이름을 변경할 수 있습니다. 즉, 행 인덱스와 열 이름 속성에 새로운 배열을 할당하는 방식으로 행 인덱스와 열 이름을 변경할 수 있음.

행 인덱스 변경: DataFrame객체.index=새로운행 인덱스 배열 열 이름 변경: DataFrame객체.columns=새로운열 이름 배열

	연령	남녀	소속
학생1	15	남	덕영중
학생2	17	여	수리중

```
In [1]: import pandas as pd

In [2]: df = pd.DataFrame([[15, '남', '덕영중'], [17, '여', '수리중']],
   ...:                   index=['준서', '예은'],
   ...:                   columns=['나이', '성별', '학교'])

In [3]: df.index=['학생1', '학생2']
   ...: df.columns=['연령', '남녀', '소속']

In [4]: print(df)
     연령 남녀   소속
학생1  15  남  덕영중
학생2  17  여  수리중
```

2) 일부 행 인덱스와 열 이름 변경

- 데이터프레임객체의 행 인덱스 또는 열 이름의 전체가 아닌 일부를 변경하고자 할 수 있음.
- rename() 함수의 index 및 columns 옵션을 통해 기존의 이름과 변경하고자 하는 이름을 딕셔너리형태로 구성하여 선택한 행 인덱스와 열 이름을 변경할 수 있음.
- rename() 함수의 경우 원본 객체를 직접 수정하는 것이 아니라 새로운 데이터프레임 객체를 반환하는 방식이므로 원본 객체를 변경하고자 한다면 inplace=True 옵션을 사용하면 됨.

행 인덱스 변경: DataFrame객체.rename(index={기존인덱스:새인덱스,…})
열 이름 변경: DataFrame객체.rename(columns={기존이름:새이름,…})

	연령	남녀	소속
준서	15	남	덕영중
예은	17	여	수리중

```
In [1]: import pandas as pd

In [2]: df = pd.DataFrame([[15, '남', '덕영중'], [17, '여', '수리중']],
   ...:                   index=['준서', '예은'],
   ...:                   columns=['나이', '성별', '학교'])

In [3]: df.rename(columns={'나이':'연령', '성별':'남녀', '학교':'소속'}, inplace=True)
   ...: df.rename(index={'학생1':'준서', '학생2':'예은' }, inplace=True)

In [4]: print(df)
    연령 남녀   소속
준서  15  남  덕영중
예은  17  여  수리중
```

○ 행과 열 삭제

1) 선택한 행과 열 삭제

- 데이터프레임의 행 또는 열을 삭제하는 명령으로 drop() 함수가 있음.
- 행을 삭제하고자 할 때는 축을 지정하는 axis 옵션에 0을 지정하고, 열

을 삭제하고자 할 때는 axis 옵션에 1을 지정하면 됨.

- 2개 이상의 행 또는 열을 삭제하고자 할 때 리스트 형태로 입력하면 매칭되는 모든 행 또는 열 데이터를 동시에 삭제함.
- drop() 함수의 경우 원본 객체를 직접 수정하는 것이 아니라 새로운 데이터프레임 객체를 반환하는 방식이므로 원본 객체를 변경하고자 한다면 inplace=True 옵션을 사용하면 됨.

행 삭제: DataFrame객체.drop(행 인덱스 또는 배열, axis=0) 열 삭제: DataFrame객체.drop(열 이름 또는 배열, axis=1)

```
In [1]: import pandas as pd

In [2]: exam_data = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
   ...:
   ...: df = pd.DataFrame(exam_data, index=['서준', '우현', '인아'])

In [3]: print(df)
    수학  영어   음악   체육
서준  90  98   85  100
우현  80  89   95   90
인아  70  95  100   90
```

```
In [1]: import pandas as pd

In [2]: exam_data = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
   ...:
   ...: df = pd.DataFrame(exam_data, index=['서준', '우현', '인아'])

In [3]: df.drop('우현', inplace=True)

In [4]: print(df)
    수학  영어   음악   체육
서준  90  98   85  100
인아  70  95  100   90
```

```
In [1]: import pandas as pd

In [2]: exam_data = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
   ...:
   ...: df = pd.DataFrame(exam_data, index=['서준', '우현', '인아'])

In [3]: df.drop(['우현', '인아'], axis=0, inplace=True)

In [4]: print(df)
    수학  영어  음악   체육
서준  90  98  85  100
```

```
In [1]: import pandas as pd

In [2]: exam_data = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
   ...:
   ...: df = pd.DataFrame(exam_data, index=['서준', '우현', '인아'])

In [3]: df.drop('수학', axis=1, inplace=True)

In [4]: print(df)
    영어   음악   체육
서준  98   85  100
우현  89   95   90
인아  95  100   90
```

```
In [1]: import pandas as pd

In [2]: exam_data = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
   ...:
   ...: df = pd.DataFrame(exam_data, index=['서준', '우현', '인아'])

In [3]: df.drop(['수학','음악'], axis=1, inplace=True)

In [4]: print(df)
    영어   체육
서준  98  100
우현  89   90
인아  95   90
```

○ 행, 열, 원소 선택

1) 행 선택

- 데이터프레임의 행 데이터를 선택하기 위해서는 loc와 iloc 인덱서를 사용합니다. 인덱스 이름을 기준으로 행을 선택할 때는 loc를 이용하고, 정수형 위치 인덱스를 사용할 때는 iloc를 이용함.
- 2개 이상의 행 인덱스를 리스트 형태로 입력하면 매칭되는 모든 행 데이

터를 동시에 추출할 수 있음.

구분	.loc	.iloc
탐색대상	인덱스 이름	정수형 위치 인덱스
범위지정	가능(범위의 끝 포함) ex) ['a':'c'] -> 'a', 'b', 'c'	가능(범위의 끝 제외) ex) [1:3] -> 1, 2
비연속 범위 지정	가능(2개의 행 인덱스 사용) ex) [['a','c']] -> 'a', 'c'	

```
In [1]: import pandas as pd

In [2]: exam_data = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
   ...: df = pd.DataFrame(exam_data, index=['서준', '우현', '인아'])

In [3]: df.loc['서준']
Out[3]:
수학     90
영어     98
음악     85
체육    100
Name: 서준, dtype: int64

In [4]: df.iloc[0]
Out[4]:
수학     90
영어     98
음악     85
체육    100
Name: 서준, dtype: int64
```

```
In [1]: import pandas as pd

In [2]: exam_data = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
   ...: df = pd.DataFrame(exam_data, index=['서준', '우현', '인아'])

In [3]: df.loc['서준':'우현']
Out[3]:
    수학  영어  음악   체육
서준  90  98  85  100
우현  80  89  95   90

In [4]: df.iloc[0:1]
Out[4]:
    수학  영어  음악   체육
서준  90  98  85  100
```

2) 열 선택

- 데이터프레임의 열 데이터를 1개만 선택할 때는 대괄호([]) 안에 열 이름을 따옴표(“”)와 함께 입력하거나, 마침표(.) 다음에 열 이름을 입력하는 두 가지 방식을 사용할 수 있음. 마침표 사용의 경우 반드시 열 이름이 문자열일 경우에만 가능함.
- 대괄호 안에 열 이름의 리스트를 입력하면 리스트의 원소인 열을 모두 선택하여 데이터프레임으로 반환함.

열 1개 선택: DataFrame객체[“열 이름”] 또는 DataFrame객체.열이름
열 n개 선택: DataFrame객체[[열1, 열2, …, 열n]]

```
In [1]: import pandas as pd

In [2]: exam_data = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
   ...: df = pd.DataFrame(exam_data, index=['서준', '우현', '인아'])

In [3]: df['수학']
Out[3]:
서준    90
우현    80
인아    70
Name: 수학, dtype: int64

In [4]: df.영어
Out[4]:
서준    98
우현    89
인아    95
Name: 영어, dtype: int64
```

```
In [1]: import pandas as pd

In [2]: exam_data = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
   ...: df = pd.DataFrame(exam_data, index=['서준', '우현', '인아'])

In [3]: df[['음악', '체육']]
Out[3]:
     음악   체육
서준   85  100
우현   95   90
인아  100   90
```

3) 원소 선택

- 데이터프레임의 행 인덱스와 열 이름을 [행, 열] 형식의 2차원 좌표로 입력하여 원소 위치를 지정하는 방법임.
- 행 선택 방법과 마찬가지로 행과 열의 이름을 기준으로 행을 선택할 때는 loc를 이용하고, 정수형 위치를 사용할 때는 iloc를 이용함.

인덱스 이름: DataFrame객체.loc[행인덱스, 열이름]
정수 위치 인덱스: DataFrame객체.iloc[행번호, 열번호]

```
In [1]: import pandas as pd

In [2]: exam_data = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
   ...: df = pd.DataFrame(exam_data, index=['서준', '우현', '인아'])

In [3]: df.loc['서준', '음악']
Out[3]: 85

In [4]: df.iloc[0, 2]
Out[4]: 85
```

```
In [1]: import pandas as pd

In [2]: exam_data = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
   ...: df = pd.DataFrame(exam_data, index=['서준', '우현', '인아'])

In [3]: df.loc['서준', ['음악', '체육']]
Out[3]:
음악     85
체육    100
Name: 서준, dtype: int64

In [4]: df.iloc[0, [2, 3]]
Out[4]:
음악     85
체육    100
Name: 서준, dtype: int64
```

○ 행과 열 추가

1) 행 추가

- 추가하려는 행 이름과 데이터 값을 loc 인덱서를 이용하여 입력합니다. 하나의 데이터 값 또는 열의 개수에 맞게 배열 형태로 여러개의 값을 입력할 수 있음.
- 하나의 값만 입력한 경우 행의 모든 원소에 같은 값이 추가됨.
- 새로운 행을 추가할 때는 기존 행 인덱스와 겹치지 않는 새로운 인덱스를 사용하고, 중복되는 경우 기존의 행의 원소값을 변경함.

DataFrame객체.loc[“새로운행 이름”] = 데이터 값 (또는 배열)

```
In [1]: import pandas as pd

In [2]: exam_data = {'이름' : ['서준', '우현', '인아'],
   ...:              '수학' : [ 90, 80, 70],
   ...:              '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100],
   ...:              '체육' : [ 100, 90, 90]}

In [3]: df = pd.DataFrame(exam_data)

In [4]: df.loc[3] = 0

In [5]: print(df)
   이름  수학  영어   음악   체육
0  서준  90  98   85  100
1  우현  80  89   95   90
2  인아  70  95  100   90
3   0   0   0    0    0
```

2) 열 추가

- 대괄호([]) 안에 추가하려는 열 이름을 따옴표(“”)와 함께 입력하고, 데이터값을 할당하는 방법으로 입력함.
- 데이터프레임의 마지막 열에 덧붙이듯 새로운 열이 추가됨.
- 만약 하나의 값만 입력한 경우 모든 행에 동일한 값이 입력됨.

DataFrame객체[“추가하려는열 이름”] = 데이터 값

```
In [1]: import pandas as pd

In [2]: exam_data = {'이름' : ['서준', '우현', '인아'],
   ...:              '수학' : [ 90, 80, 70],
   ...:              '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100],
   ...:              '체육' : [ 100, 90, 90]}
   ...: df = pd.DataFrame(exam_data)

In [3]: df['국어'] = 80

In [4]: print(df)
   이름  수학  영어   음악   체육  국어
0  서준  90  98   85  100  80
1  우현  80  89   95   90  80
2  인아  70  95  100   90  80
```

○ 변경

1) 원소 값 변경

- 데이터프레임의 특정 원소를 선택하고 새로운 데이터값을 지정해주면 원소값이 변경됨.
- 원소 1개를 선택하여 변경할 수도 있고, 여러 개의 원소를 선택하여 한꺼번에 값을 바꿀 수도 있음.

DataFrame객체의 일부분 또는 원소 선택=새로운 값

```
In [1]: import pandas as pd

In [2]: exam_data = {'이름' : ['서준', '우현', '인아'],
   ...:              '수학' : [ 90, 80, 70],
   ...:              '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100],
   ...:              '체육' : [ 100, 90, 90]}
   ...: df = pd.DataFrame(exam_data)

In [3]: df.iloc[0,3] = 80

In [4]: print(df)
   이름  수학  영어   음악   체육
0  서준  90  98   80  100
1  우현  80  89   95   90
2  인아  70  95  100   90
```

2) 행과 열 위치 변경 변경

- 데이터프레임의 행과 열을 서로 맞바꾸는 방법으로 전치행렬과 같은 개념임.

DataFrame객체.transpose() 또는 DataFrame객체.T

```
In [1]: import pandas as pd

In [2]: exam_data = {'이름' : ['서준', '우현', '인아'],
   ...:              '수학' : [ 90, 80, 70],
   ...:              '영어' : [ 98, 89, 95],
   ...:              '음악' : [ 85, 95, 100],
   ...:              '체육' : [ 100, 90, 90]}
   ...: df = pd.DataFrame(exam_data)

In [3]: df.transpose()
Out[3]:
      0   1   2
이름   서준  우현   인아
수학   90  80   70
영어   98  89   95
음악   85  95  100
체육  100  90   90
```

○ 데이터프레임 정렬

1) 행 인덱스를 기준으로 데이터프레임 정렬

- sort_index() 함수를 활용하여 행 인덱스를 기준으로 데이터프레임의 값을 정렬할 수 있음.
- 여기서 ascending 옵션을 사용하여 오름차순(True) 또는 내림차순(False)을 설정할 수 있음.

DataFrame객체.sort_index(ascending=True/False)

```
In [1]: import pandas as pd

In [2]: dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9], 'c3':[10,11,12], 'c4':[13,14,15]}
   ...: df = pd.DataFrame(dict_data, index=['r0', 'r1', 'r2'])

In [3]: print(df)
    c0  c1  c2  c3  c4
r0   1   4   7  10  13
r1   2   5   8  11  14
r2   3   6   9  12  15

In [4]: df.sort_index(ascending=False)
Out[4]:
    c0  c1  c2  c3  c4
r2   3   6   9  12  15
r1   2   5   8  11  14
r0   1   4   7  10  13
```

2) 특정 열의 데이터 값을 기준으로 데이터프레임 정렬

- sort_values() 함수를 활용하여 특정 열을 기준으로 데이터프레임의 값을 정렬할 수 있음.
- 여기서 by 옵션을 사용하여 특정 열을 지정할 수 있고, ascending 옵션을 사용하여 오름차순(True) 또는 내림차순(False)을 설정할 수 있음.

DataFrame객체.sort_values(by=기준열이름, ascending=True/False)

```
In [1]: import pandas as pd

In [2]: dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9], 'c3':[10,11,12], 'c4':[13,14,15]}
   ...: df = pd.DataFrame(dict_data, index=['r0', 'r1', 'r2'])

In [3]: print(df)
    c0  c1  c2  c3  c4
r0   1   4   7  10  13
r1   2   5   8  11  14
r2   3   6   9  12  15

In [4]: df.sort_values(by='c1', ascending=False)
Out[4]:
    c0  c1  c2  c3  c4
r2   3   6   9  12  15
r1   2   5   8  11  14
r0   1   4   7  10  13
```

3. 데이터프레임 생성 및 변환 실습

1) 주어진 자료 데이터프레임 생성

※ 다음의 표를 딕셔너리 자료로 만든 후 데이터프레임으로 변환하세요.

이름	나이	직업
철수	12	중학생
민지	21	대학생
영미	18	고등학생
한솔	26	회사원

```
In [1]: import pandas as pd

In [2]: df = {"이름":["철수","민지", "영미", "한솔"], "나이":[12,21,18,26],"직업":["중학생","대학생","고등학생","회사원"]}
   ...: df1 = pd.DataFrame(df)

In [3]: print(df1)
   이름  나이    직업
0  철수  12   중학생
1  민지  21   대학생
2  영미  18  고등학생
3  한솔  26   회사원
```

1) 주어진 자료 데이터프레임 변환

※ 생성한 데이터프레임의 열 이름 중 나이를 연령으로 수정하세요.

```
In [4]: df1.rename(columns={"나이":"연령"},inplace=True)

In [5]: print(df1)
   이름  연령    직업
0  철수  12   중학생
1  민지  21   대학생
2  영미  18  고등학생
3  한솔  26   회사원
```

※ 생성한 데이터프레임의 열 이름 중 직업을 제거하세요.

```
In [8]: df1.drop("직업",axis=1,inplace=True)

In [9]: print(df1)
   이름  연령
0  철수  12
1  민지  21
2  영미  18
3  한솔  26
```

※ 생성한 데이터프레임의 민지의 연령을 선택하여 출력하세요. 그리고 이름을 선택하여 출력하세요.

```
In [10]: df1.iloc[1,1]
Out[10]: 21

In [11]: df1.이름
Out[11]:
0    철수
1    민지
2    영미
3    한솔
Name: 이름, dtype: object
```

※ 생성한 데이터프레임을 연령에 따라서 내름차순으로 정렬하세요.

```
In [13]: df1.sort_values(by="연령",ascending=False)
Out[13]:
   이름  연령
3  한솔  26
1  민지  21
2  영미  18
0  철수  12
```