In [1]:
```
# pandas, numpy moudle 불러오기
import pandas as pd
import numpy as np
```

6. 계층 색인 (다중 색인)

2개 이상의 색인(인덱스)를 지정할 수 있다.

차원이 높은 (고차원) 데이터를 낮은 차원의 형식으로 다룰 수 있게 해주는 기능

In [2]:
```
# 샘플 데이터 생성
# random.seed(0) - 8:2, random.sedd(1) - 2:8
# randint: 50에서 100사이의 수를 5행 4열로 대입
# 계층색인 column: [[], []]
np.random.seed(0)

df = pd.DataFrame(np.random.randint(50, 100, (5, 4)),
                  columns=[[2016, 2016, 2017, 2017], ['영어','수학','영어','수학']]
                  index = ['Kim','Park','Lee','Jung','Moon'])
```

In [3]:
```
df
```

Out[3]:

	2016		2017	
	영어	수학	영어	수학
Kim	94	97	50	53
Park	53	89	59	69
Lee	71	86	73	56
Jung	74	74	62	51
Moon	88	89	73	96

In [4]:
```
# index 확인
df.index
```

Out[4]:
```
Index(['Kim', 'Park', 'Lee', 'Jung', 'Moon'], dtype='object')
```

In [5]:
```
# column 확인
df.columns
```

Out[5]:
```
MultiIndex([(2016, '영어'),
            (2016, '수학'),
            (2017, '영어'),
            (2017, '수학')],
           )
```

인덱싱

다중 인덱스의 경우 여러 인덱스를 () 튜플의 형태로 묶어주어서 표현

1. xs() method - 열

한개의 column만 선택 가능

axis=0, level=0 (default)

axis=0(행), axis=1(열)

level=0(상위레벨), level=1(하위레벨)

In [6]:
```
#2016년 영어, 수학 성적 조회
# default 0 값은 생략 가능
df.xs(2016, axis = 1, level = 0)
```

Out[6]:

	영어	수학
Kim	94	97
Park	53	89
Lee	71	86
Jung	74	74
Moon	88	89

2. 기존 index loc 방식 - 열

In [7]:
```
#2016년 영어, 수학 성적 조회
df[2016]
```

Out[7]:

	영어	수학
Kim	94	97
Park	53	89
Lee	71	86
Jung	74	74
Moon	88	89

In [8]:
```
#2016년 영어 성적만 조회
#최상위 인덱스부터 명시하며, 인덱싱하고자하는 색인들을 튜플 형태로 정의
df[(2016, '영어')]
```

Out[8]:
```
Kim     94
Park    53
Lee     71
Jung    74
Moon    88
Name: (2016, 영어), dtype: int64
```

In [9]:
```
# 2017년 수학 조회
df[(2017, '수학')]
```

Out[9]:
```
Kim     53
Park    69
Lee     56
Jung    51
Moon    96
Name: (2017, 수학), dtype: int64
```

In [11]:
```
# 2017년 Kim의 수학 점수 조회
# iloc: 숫자, loc: 문자 - 계층색인에서는 주로 loc 사용
df.loc['Kim', (2017, '수학')]
```

Out[11]:
```
53
```

1. xs() method - 행

In [12]:
```
# Kim의 성적만 선택
df.xs('Kim', axis = 0)
```

Out[12]:
```
2016  영어    94
      수학    97
2017  영어    50
      수학    53
Name: Kim, dtype: int64
```

2. 기본 index 방식 - 행

In [13]:
```
# Kim의 성적만 선택
df.loc['Kim']
```

Out[13]:
```
2016  영어    94
      수학    97
2017  영어    50
      수학    53
Name: Kim, dtype: int64
```

실습

In [23]:
```
# 실습 1
# Kim, Park, Lee의 성적만 선택
df.loc[['Kim', 'Park', 'Lee']]
```

Out[23]:

	2016		2017	
	영어	수학	영어	수학
Kim	94	97	50	53
Park	53	89	59	69
Lee	71	86	73	56

In [24]:
```
#  실습 2
# 'Kim', 'Lee'의 2016년 영어 성적 조회
df.loc[['Kim', 'Lee'], (2016, '영어')]
```

Out[24]:
```
Kim    94
Lee    71
Name: (2016, 영어), dtype: int64
```

In [26]:
```
# 실습 3
# 2016, 2017년도 영어 성적만 선택
df.xs('영어', axis = 1, level = 1)
```

Out[26]:

	2016	2017
Kim	94	50
Park	53	59
Lee	71	73
Jung	74	62
Moon	88	73

In [27]:
```
# 실습 3 - 다른 풀이
df[[(2016, '영어'), (2017, '영어')]]
```

Out[27]:

	2016	2017
	영어	영어
Kim	94	50
Park	53	59
Lee	71	73
Jung	74	62
Moon	88	73

index 이름 부여하기 (set_names)

In [29]:
```
# row index의 이름을 '학생명'이라고 정의하기
df.index.set_names('학생명', inplace = True)
df
```

Out[29]:

	2016		2017	
	영어	수학	영어	수학
학생명				
Kim	94	97	50	53
Park	53	89	59	69
Lee	71	86	73	56
Jung	74	74	62	51
Moon	88	89	73	96

In [30]:
```
# column index의 이름을 '연도', '과목'이라고 정의하기
df.columns.set_names(['연도', '과목'], inplace = True)
df
```

Out[30]:

연도	2016		2017	
과목	영어	수학	영어	수학
학생명				
Kim	94	97	50	53
Park	53	89	59	69
Lee	71	86	73	56
Jung	74	74	62	51
Moon	88	89	73	96

1) swaplevel(index1, index2, axis)

index1과 index2의 위치를 변경함.

index1과 index2가 row index 경우, axis = 0

index1과 index2가 column index 경우, axis = 1

In [31]:
```
# 년도와 과목의 위치를 변경
# 상위레벨과 하위레벨 위치 이동
df.swaplevel('연도', '과목', axis = 1)
```

Out[31]:

과목	영어	수학	영어	수학
연도	2016	2016	2017	2017
학생명				
Kim	94	97	50	53
Park	53	89	59	69
Lee	71	86	73	56
Jung	74	74	62	51
Moon	88	89	73	96

2) stack(), unstack() 함수

stack(level) : 컬럼 인덱스를 로우 인덱스로 옮길 때 사용.

unstack(level): 로우 인덱스를 컬럼 인덱스로 옮길 때 사용.

level 인자는 옮기고자 하는 인덱스의 위치를 표기함. 명시하지 않은 경우, 최하단의 인덱스를 이동시킴.

level은 최상위가 0이고, 1씩 증가함

In [32]:
```
df
```

Out[32]:

연도	2016		2017	
과목	영어	수학	영어	수학
학생명				
Kim	94	97	50	53
Park	53	89	59	69
Lee	71	86	73	56
Jung	74	74	62	51
Moon	88	89	73	96

In [33]:
```
# 컬럼 인덱스 과목을 로우 인덱스로 변경하고 df2에 저장
df2 = df.stack(level = 1)
df2
```

Out[33]:

	연도	2016	2017
학생명	과목		
Kim	수학	97	53
	영어	94	50
Park	수학	89	69
	영어	53	59
Lee	수학	86	56
	영어	71	73
Jung	수학	74	51
	영어	74	62
Moon	수학	89	96
	영어	88	73

df2를 대상으로 아래 실습 문제 수행

In [34]:
```
# 실습1
# Kim의 성적만 선택
df2.loc['Kim']
```

Out[34]:

연도	2016	2017
과목		
수학	97	53
영어	94	50

In [35]:
```
# 실습2
# Park의 수학 성적만 선택
df2.loc[('Park', '수학'), :]
```

Out[35]:
```
연도
2016    89
2017    69
Name: (Park, 수학), dtype: int64
```

In [44]:
```
# 실습2 - 다른 풀이
df2.loc['Park', '수학']
```

Out[44]:
```
연도
2016    89
2017    69
Name: (Park, 수학), dtype: int64
```

In [38]:
```
# 실습3
# 모든 학생들의 영어 성적만 선택
df2.xs('영어', axis = 0, level = 1)
```

Out[38]:

연도	2016	2017
학생명		
Kim	94	50
Park	53	59
Lee	71	73
Jung	74	62
Moon	88	73

In [40]:
```
# 실습4
# Park 학생의 2016년 영어 성적만 출력
df2.loc[('Park', '영어'), 2016]
```

Out[40]:
```
53
```

In [45]:
```
# 실습5
# 학생들의 과목별 성적의 평균을 구해서, 새로운 컬럼 '평균'으로 저장
df2['평균'] = df2.mean(axis = 1)
df2
```

Out[45]:

	연도	2016	2017	평균
학생명	과목			
Kim	수학	97	53	75.0
	영어	94	50	72.0
Park	수학	89	69	79.0
	영어	53	59	56.0
Lee	수학	86	56	71.0
	영어	71	73	72.0
Jung	수학	74	51	62.5
	영어	74	62	68.0
Moon	수학	89	96	92.5
	영어	88	73	80.5

실습

data/NC Dinos.xlsx 파일을 읽어서, 아래 결과처럼 나오도록 하시오.

	안타			홈런		
년도	2013	2014	2015	2013	2014	2015
선수명						
강구성	0	-	1	0	-	0
강민국	-	0	0	-	0	0
강진성	1	-	-	0	-	-
권희동	-	63	-	-	7	-
김동건	2	-	-	1	-	-
김성욱	1	4	-	0	1	-
김종찬	1	-	-	0	-	-
김종호	129	-	125	0	-	4
김준완	-	2	10	-	0	0
김태군	-	-	107	-	-	6
김태우	-	1	-	-	0	-
김태진	-	-	0	-	-	0
나성범	98	157	184	14	30	28

In [ ]:
```
# 데이터 불러오기

# 필요없는 컬럼 제거


# 연도 추가

# index 설정
```