판다스 조작하기 1

이준원
SKT Data Scientist

학습 목표

1. 판다스 시리즈에 대해 설명할 수 있다.
2. 판다스 데이터프레임에 대해 설명할 수 있다.
3. 데이터프레임의 기초 조작을 할 수 있다.

K-Digital 훈련원

CONTENTS

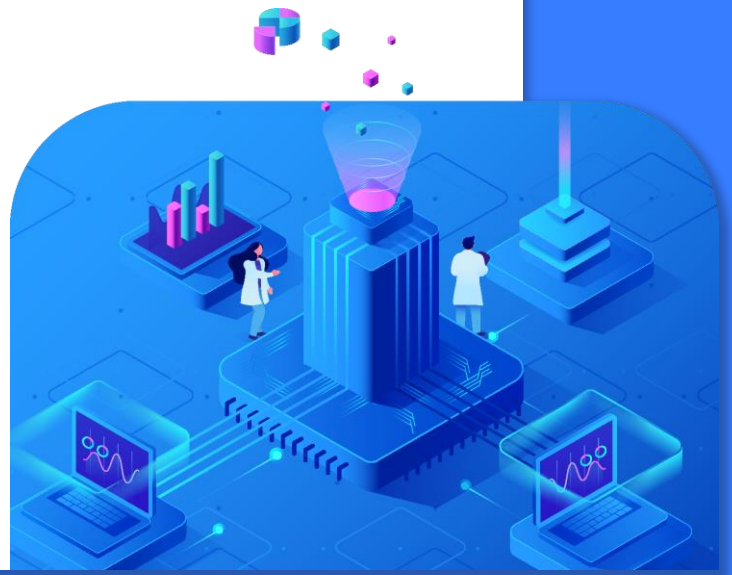
I

판다스 시리즈

CHAPTER

- 판다스란
- 시리즈 및 데이터 프레임

Institute for
K-Digital Training
미래를 향한
인재를 양성합니다



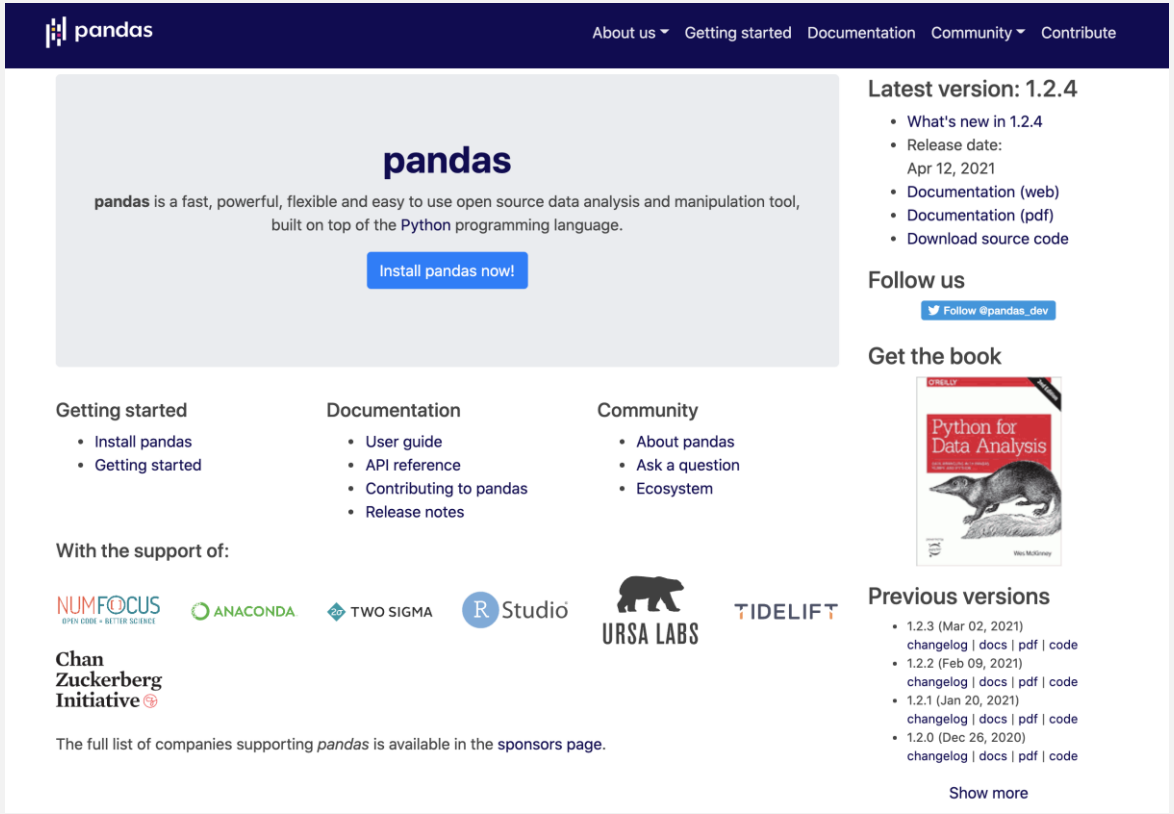
판다스란 무엇인가?

- 데이터를 수집하고 정리하는 데 최적화된 도구
- 오픈소스로 무료라는 장점을 가짐
- 가장 쉬운 언어인 파이썬을 기반으로 함
- 데이터를 다루는 업무의 80% 이상을 판다스로 처리함

➔ 데이터 과학자에게 가장 기본적이면서 아주 중요한 도구!

판다스란 무엇인가?

- 판다스 공식 홈페이지 (https://pandas.pydata.org/)



판다스 자료구조

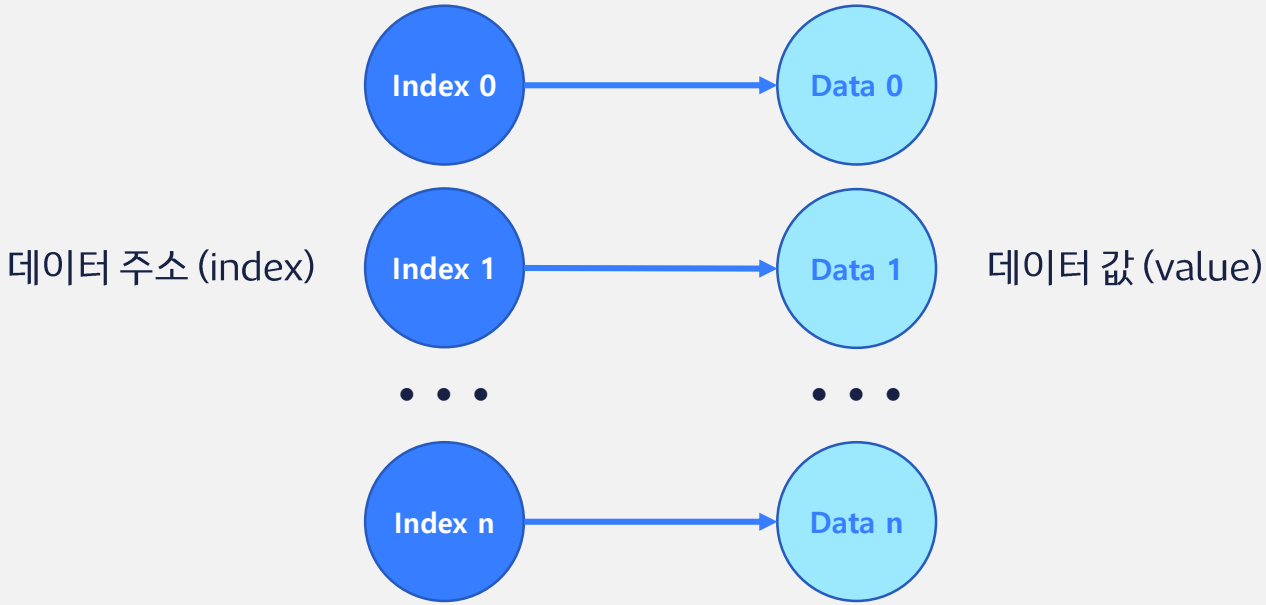
시리즈와 데이터 프레임

- 데이터의 속성은 매우 다양함
  - 서로 다른 형식의 데이터를 동일한 형식으로 통합해야 함
  - 판다스는 시리즈와 데이터 프레임을 제공함
- 시리즈
  - 1차원 배열
- 데이터 프레임
  - 2차원 배열
  - 행과 열로 이루어진 데이터 (매트릭스)

시리즈 (Series)

- 데이터가 순차적으로 나열된 1차원 배열
- 인덱스(key)와 데이터 값(value)가 일대일 대응로 이루어진 데이터

➔ 파이썬 딕셔너리 {key: value}와 비슷함



판다스란 무엇인가?

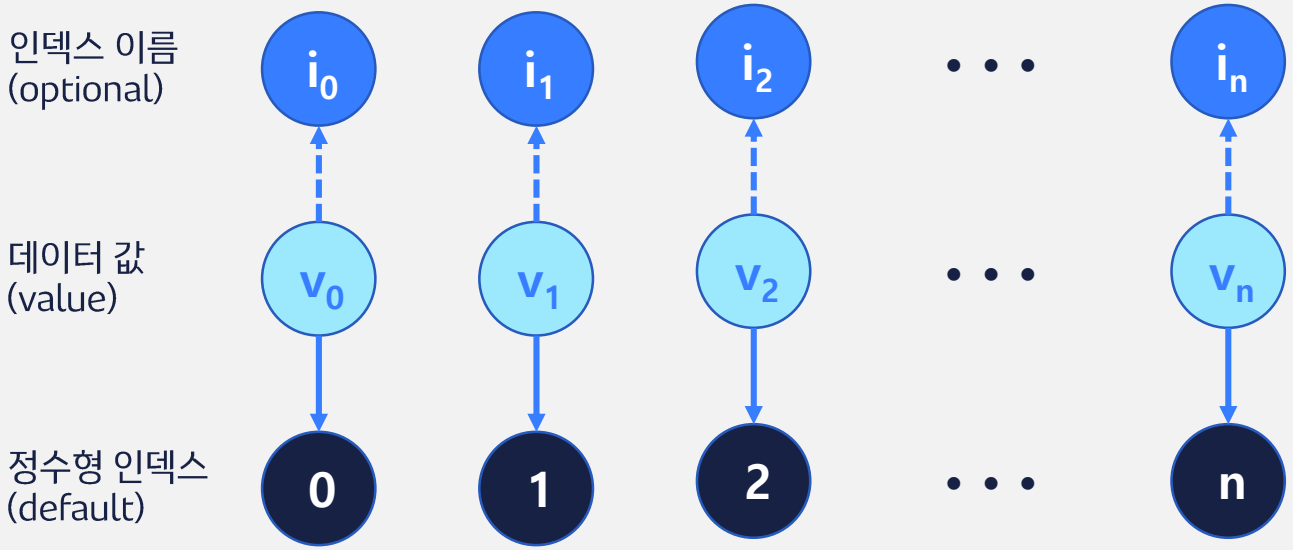
- 파이썬 딕셔너리를 시리즈로 변환

```
'''
    딕셔너리 => 시리즈 변환
    pandas.Series(딕셔너리)
'''

import pandas as pd

dict_data = {'a':1, 'b':2, 'c':3}

series = pd.Series(dict_data)

print(type(series))
print(series)
```

```
<class 'pandas.core.series.Series'>
a    1
b    2
c    3
dtype: int64
```

인덱스 구조

- 인덱스를 잘 활용하면 값의 탐색, 정렬, 선택, 결합을 쉽게 할 수 있음
- 정수형 인덱스가 기본으로 사용되며, 직접 이름을 지정할 수도 있음



인덱스 구조

- 인덱스(index)와 값(value)로 구성되어 있음

```
'''
    인덱스 배열: Series.index
    데이터 값 배열: Series.values
'''
import pandas as pd

lis_data = ['2021-04-26', '철수', '남', 100, True]
series = pd.Series(lis_data)

print(series)
print('\n')
print(series.index)
print('\n')
print(series.values)
```

```
0    2021-04-26
1            철수
2             남
3           100
4          True
dtype: object


RangeIndex(start=0, stop=5, step=1)


['2021-04-26' '철수' '남' 100 True]
```

시리즈 원소 선택

```
import pandas as pd

lis_data = ['2021-04-26', '철수', '남', 100, True]
index_names = ['날짜', '이름', '성별', '점수', '전학여부']

series = pd.Series(lis_data, index = index_names)

print(series)
print('\n')
print(series[0], series['이름'])
print('\n')
print(series[[0, 2]])
print('\n')
print(series[['날짜', '전학여부']])
```

```
날짜       2021-04-26
이름               철수
성별                남
점수              100
전학여부           True
dtype: object


2021-04-26 철수


날짜     2021-04-26
성별               남
dtype: object


날짜       2021-04-26
전학여부           True
dtype: object
```

시리즈 원소 선택

```
print(series)
print('\n')
print(series[2:5])
print('\n')
print(series['성별':'전학여부'])
```

```
날짜       2021-04-26
이름               철수
성별                남
점수               100
전학여부            True
dtype: object


성별          남
점수        100
전학여부     True
dtype: object


성별          남
점수        100
전학여부     True
dtype: object
```

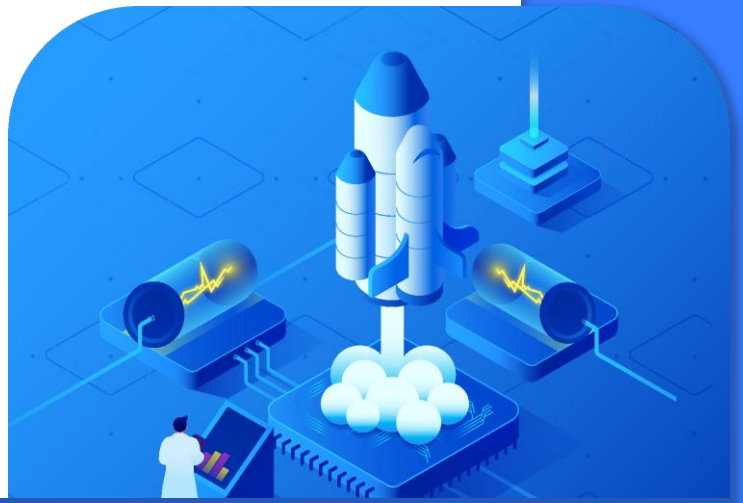
II

판다스 데이터프레임

CHAPTER

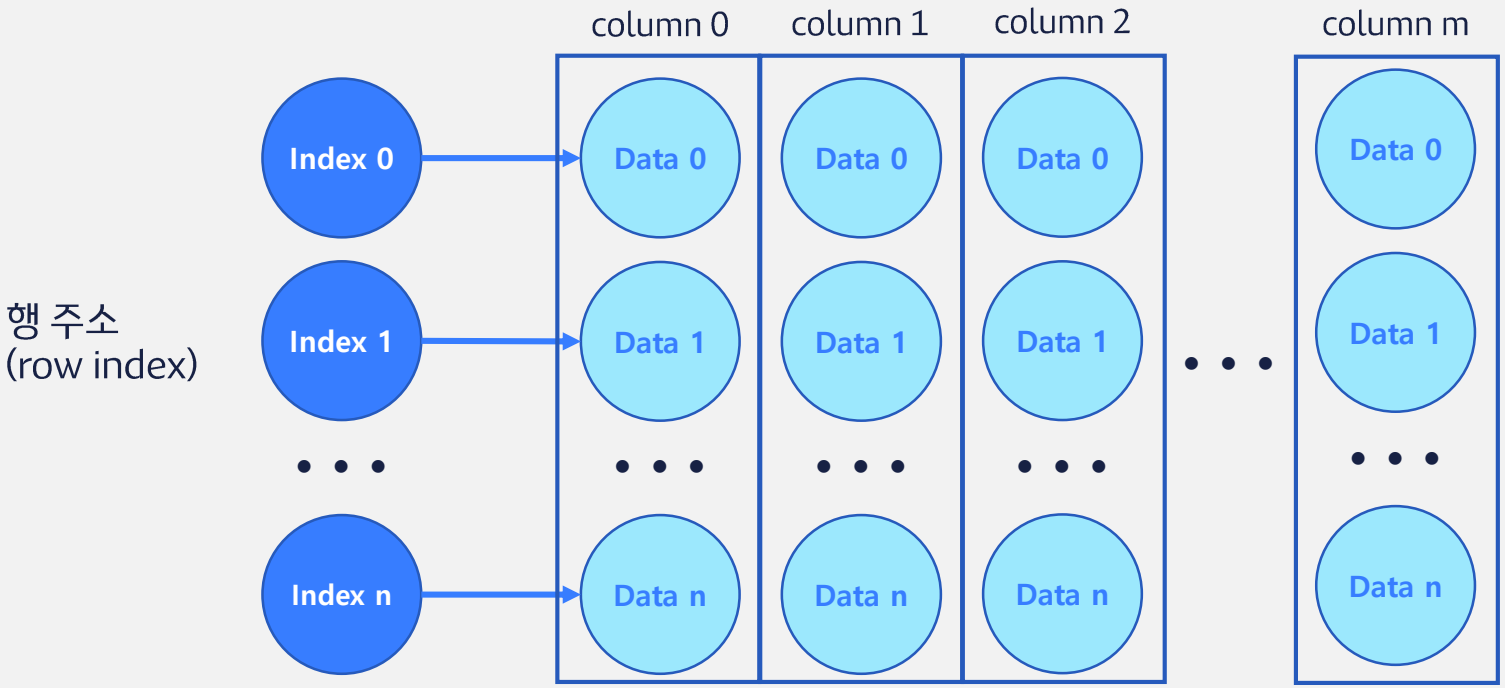
- 데이터프레임 생성
- 행/열 활용

Institute for
K-Digital Training
미래를 향한
인재를 양성합니다



데이터프레임 (DataFrame)

- 2차원 배열로 행과 열로 이루어짐
- 여러 개의 시리즈들이 모여서 데이터 프레임을 이룸



데이터프레임 만들기

```
import pandas as pd

dic_data = {
    'col1': [0,1,2],
    'col2': [3,4,5],
    'col3': [6,7,8],
    'col4': [9,10,11]
}

df = pd.DataFrame(dic_data)

print(type(df))
df
```

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

	col1	col2	col3	col4
0	0	3	6	9
1	1	4	7	10
2	2	5	8	11

- 딕셔너리를 사용하여 데이터프레임을 만들 수 있음

행/열 이름 설정

```
import pandas as pd

df = pd.DataFrame(
    [['남', 30, '경기고'], ['여', 25, '서울고']],
    index = ['기범', '혜리'],
    columns = ['gender', 'age', 'school']
)

print(df)
print('\n')
print(df.index)
print('\n')
print(df.values)
```

```
   gender  age school
기범      남   30    경기고
혜리      여   25    서울고


Index(['기범', '혜리'], dtype='object')


[['남' 30 '경기고']
 ['여' 25 '서울고']]
```

- index: 행의 이름
- columns: 열의 이름

K-Digital 훈련원

행/열 이름 설정

```
print(df)
```

```
   gender  age school
기범      남   30    경기고
혜리      여   25    서울고
```

```
df.index = ['이름1', '이름2']
df.columns = ['성별', '연령', '학교']
print(df)
```

```
     성별  연령   학교
이름1  남   30  경기고
이름2  여   25  서울고
```

```
df.rename(columns = {'gender':'성별', 'age':'연령', 'school':'학교'})
df.rename(index = {'기범':'이름1', '혜리': '이름2'})
print(df)
```

```
     성별  연령   학교
이름1  남   30  경기고
이름2  여   25  서울고
```

- 행(index)과 열(columns)의 이름을 임의대로 설정할 수 있음

행 선택

구분	loc 메소드	iloc 메소드
탐색 대상	인덱스 이름으로 탐색	정수형 위치 인덱스로 탐색
범위 지정	['a':'c'] ➔ ['a', 'b', 'c']	[3:7] ➔ 3, 4, 5, 6 (7 제외)

행 선택

```
exam_data = {
    '국어':[100, 80, 30], '수학':[90, 80, 50],
    '영어':[100, 40, 90], '음악':[80, 100, 40]
}
df = pd.DataFrame(exam_data, index = ['기범', '혜리', '동엽'])

print(df)
print('\n')

location = df.loc[['기범','혜리']]
print(location)
print('\n')

ilocation = df.iloc[[1, 2]]
print(ilocation)
```

```
     국어  수학   영어   음악
기범  100  90  100   80
혜리   80  80   40  100
동엽   30  50   90   40


     국어  수학   영어   음악
기범  100  90  100   80
혜리   80  80   40  100


    국어  수학  영어   음악
혜리  80  80  40  100
동엽  30  50  90   40
```

- loc 메소드는 인덱스의 이름으로 데이터를 선택
- iloc 메소드는 인덱스의 위치로 데이터를 선택

iloc 고급 활용

```
exam_data = {
    '국어':[100, 80, 30, 90], '수학':[90, 80, 50, 30],
    '영어':[100, 40, 90, 80], '음악':[80, 100, 40, 50]
}
df = pd.DataFrame(exam_data, index = ['기범', '혜리', '동엽', '나래'])

print(df)
print('\n')

df1 = df.iloc[0:3:2]
print(df1)
print('\n')

df2 = df.iloc[::-1]
print(df2)
```

```
     국어  수학   영어   음악
기범  100  90  100   80
혜리   80  80   40  100
동엽   30  50   90   40
나래   90  30   80   50


     국어  수학   영어  음악
기범  100  90  100  80
동엽   30  50   90  40


     국어  수학   영어   음악
나래   90  30   80   50
동엽   30  50   90   40
혜리   80  80   40  100
기범  100  90  100   80
```

- iloc을 활용하여 범위를 지정하고, 슬라이싱 간격을 조정할 수 있음
- 슬라이싱 간격에 -1을 입력하면 역순으로 정렬됨

열 선택

```
exam_data = {
    '국어':[100, 80, 30], '수학':[90, 80, 50],
    '영어':[100, 40, 90], '음악':[80, 100, 40]
}
df = pd.DataFrame(exam_data, index = ['기범', '혜리', '동엽'])

print(df)
print('\n')

col1 = df['국어']
print(col1)
print('\n')

df1 = df[['영어', '음악']]
print(df1)
```

```
     국어  수학   영어   음악
기범  100  90  100   80
혜리   80  80   40  100
동엽   30  50   90   40


기범    100
혜리     80
동엽     30
Name: 국어, dtype: int64


     영어   음악
기범  100   80
혜리   40  100
동엽   90   40
```

- 열을 하나 선택할 때는 열의 이름으로 선택
- 2개 이상 선택할 때는 []를 사용한 열 이름의 리스트로 선택

원소 선택

```
exam_data = {
    '국어':[100, 80, 30, 90], '수학':[90, 80, 50, 30],
    '영어':[100, 40, 90, 80], '음악':[80, 100, 40, 50]
}
df = pd.DataFrame(exam_data, index = ['기범', '혜리', '동엽', '나래'])

print(df)
print('\n')

elem1 = df.loc['혜리', '영어']
print('혜리의 영어 점수:', elem1)

elem2 = df.iloc[0, 3]
print('기범(0)의 음악(3) 점수', elem2)
```

```
     국어  수학   영어   음악
기범  100  90  100   80
혜리   80  80   40  100
동엽   30  50   90   40
나래   90  30   80   50


혜리의 영어 점수: 40
기범(0)의 음악(3) 점수 80
```

- loc 메소드로 원소를 선택할 때는, 행과 열의 이름으로 선택
- iloc 메소드로 원소를 선택할 때는 행과 열의 인덱스를 계산하여 선택

III

데이터프레임 조작

CHAPTER

- 행/열 조작
- 데이터프레임 정렬

Institute for
K-Digital Training
미래를 향한
인재를 양성합니다

데이터프레임 조작

- 주어진 데이터프레임에 새로운 데이터를 추가하거나 삭제할 수 있음
  - 행(row) 추가/삭제
  - 열(column) 추가/삭제

- 데이터프레임 내 특정 값이 잘못될 경우 이를 변경할 수 있음

- 데이터프레임의 행과 열의 위치를 바꿔서 조작할 수 있음

- 데이터프레임 내 행 데이터를 특정 기준에 따라 정렬할 수 있음

행 추가

```
exam_data = {
    '이름':['기범', '혜리', '동엽'],
    '국어':[100, 80, 30], '수학':[90, 80, 50],
    '영어':[100, 40, 90], '음악':[80, 100, 40]
}
df = pd.DataFrame(exam_data)

print(df)
print('\n')

df.loc[4] = ['동현', 80, 90, 30, 40]
print(df)
print('\n')

df.loc[5] = df.loc[0]
print(df)
```

```
   이름   국어  수학   영어   음악
0  기범  100  90  100   80
1  혜리   80  80   40  100
2  동엽   30  50   90   40


   이름   국어  수학   영어   음악
0  기범  100  90  100   80
1  혜리   80  80   40  100
2  동엽   30  50   90   40
4  동현   80  90   30   40


   이름   국어  수학   영어   음악
0  기범  100  90  100   80
1  혜리   80  80   40  100
2  동엽   30  50   90   40
4  동현   80  90   30   40
5  기범  100  90  100   80
```

- loc 메소드에 행의 이름과 데이터를 전달하여 새로운 행을 추가함
- 열의 개수에 맞게 데이터를 추가해야 함
- 기존의 행의 이름을 선택하면 새로 데이터가 추가되지 않고 기존 데이터가 변경됨

열 추가

```
exam_data = {
    '국어':[100, 80, 30, 90], '수학':[90, 80, 50, 30],
    '영어':[100, 40, 90, 80], '음악':[80, 100, 40, 50]
}
df = pd.DataFrame(exam_data, index = ['기범', '혜리', '동엽', '나래'])

print(df)
print('\n')

df['미술'] = 70
print(df)
print('\n')

df['체육'] = [10, 20, 30, 40]
print(df)
```

```
    국어  수학   영어   음악
기범  100  90  100   80
혜리   80  80   40  100
동엽   30  50   90   40
나래   90  30   80   50


    국어  수학   영어   음악  미술
기범  100  90  100   80  70
혜리   80  80   40  100  70
동엽   30  50   90   40  70
나래   90  30   80   50  70


    국어  수학   영어   음악  미술  체육
기범  100  90  100   80  70  10
혜리   80  80   40  100  70  20
동엽   30  50   90   40  70  30
나래   90  30   80   50  70  40
```

- 새로운 열의 이름과 데이터 값(리스트)을 사용하여 열을 추가
- 하나의 값만 전달하면 모두 같은 값으로 채워짐
- 기존의 열의 이름을 사용하면 값이 채워지지 않고 해당 열의 데이터가 변경됨

행 삭제

- drop 메소드를 사용하여 해당 이름의 행을 삭제함 (axis = 0)

```
import pandas as pd

exam_data = {
    '국어':[100, 80, 30], '수학':[90, 80, 50],
    '영어':[100, 40, 90], '음악':[80, 100, 40]
}
df = pd.DataFrame(exam_data, index = ['기범', '혜리', '동엽'])

print(df)
print('\n')

df2 = df[:]
df2.drop('기범', inplace = True)
print(df2)
print('\n')

df3 = df[:]
df3.drop(['기범', '동엽'], axis = 0, inplace = True)
print(df3)
```

```
     국어  수학   영어   음악
기범  100  90  100   80
혜리   80  80   40  100
동엽   30  50   90   40


    국어  수학  영어   음악
혜리  80  80  40  100
동엽  30  50  90   40


    국어  수학  영어   음악
혜리  80  80  40  100
```

열 삭제

- axis = 1, drop 메소드를 사용하면 해당 이름을 가진 열이 삭제됨

```
exam_data = {
    '국어':[100, 80, 30], '수학':[90, 80, 50],
    '영어':[100, 40, 90], '음악':[80, 100, 40]
}
df = pd.DataFrame(exam_data, index = ['기범', '혜리', '동엽'])

print(df)
print('\n')

df2 = df[:]
df2.drop('국어', axis = 1, inplace = True)
print(df2)
print('\n')

df3 = df[:]
df3.drop(['국어', '음악'], axis = 1, inplace = True)
print(df3)
```

```
     국어  수학   영어   음악
기범  100  90  100   80
혜리   80  80   40  100
동엽   30  50   90   40


     수학   영어   음악
기범  90  100   80
혜리  80   40  100
동엽  50   90   40


     수학   영어
기범  90  100
혜리  80   40
동엽  50   90
```

원소값 변경

- 행과 열의 이름을 사용하여 loc 메소드로 변경
- 행과 열의 인덱스 위치를 사용하여 iloc 메소드로 변경

```
exam_data = {
    '국어':[100, 80, 30, 90], '수학':[90, 80, 50, 30],
    '영어':[100, 40, 90, 80], '음악':[80, 100, 40, 50]
}
df = pd.DataFrame(exam_data, index = ['기범', '혜리', '동엽', '나래'])

print(df)
print('\n')

df.loc['혜리', '영어'] = 80
df.iloc[0, 3] = 30

print(df)
```

```
    국어  수학   영어   음악
기범  100  90  100   80
혜리   80  80   40  100
동엽   30  50   90   40
나래   90  30   80   50


    국어  수학   영어   음악
기범  100  90  100   30
혜리   80  80   80  100
동엽   30  50   90   40
나래   90  30   80   50
```

행/열 위치 바꾸기 (transpose)

```
exam_data = {
    '국어':[100, 80, 30, 90], '수학':[90, 80, 50, 30],
    '영어':[100, 40, 90, 80], '음악':[80, 100, 40, 50]
}
df = pd.DataFrame(exam_data, index = ['기범', '혜리', '동엽', '나래'])

print(df)
print('\n')

df = df.transpose()
print(df)
print('\n')

df = df.T
print(df)
```

```
     국어  수학   영어   음악
기범  100  90  100   80
혜리   80  80   40  100
동엽   30  50   90   40
나래   90  30   80   50


     기범   혜리  동엽  나래
국어  100   80  30  90
수학   90   80  50  30
영어  100   40  90  80
음악   80  100  40  50


     국어  수학   영어   음악
기범  100  90  100   80
혜리   80  80   40  100
동엽   30  50   90   40
나래   90  30   80   50
```

- 2차원 numpy의 transpose 동일하게 작동
- transpose(), T 모두 사용 가능

데이터프레임 정렬

- index의 순서대로 정렬 → sort_index()

```
import pandas as pd

dic_data = {
    'col1': [0,1,2],
    'col2': [5,4,3],
    'col3': [6,7,8],
    'col4': [9,11,10]
}

df = pd.DataFrame(dic_data, index = ['row1', 'row2', 'row3'])
print(df)
print('\n')

idf = df.sort_index(ascending=False)
print(idf)
```

```
      col1  col2  col3  col4
row1     0     5     6     9
row2     1     4     7    11
row3     2     3     8    10


      col1  col2  col3  col4
row3     2     3     8    10
row2     1     4     7    11
row1     0     5     6     9
```

데이터프레임 정렬

- 특정 value의 순서대로 정렬 → sort_value()

```
import pandas as pd

dic_data = {
    'col1': [0,1,2],
    'col2': [5,4,3],
    'col3': [6,7,8],
    'col4': [9,11,10]
}

df = pd.DataFrame(dic_data, index = ['row1', 'row2', 'row3'])
print(df)
print('\n')

idf = df.sort_index(ascending=False)
print(idf)
```

```
      col1  col2  col3  col4
row1     0     5     6     9
row2     1     4     7    11
row3     2     3     8    10


      col1  col2  col3  col4
row3     2     3     8    10
row2     1     4     7    11
row1     0     5     6     9
```

내용 정리

- 판다스는 2가지 자료구조가 존재함
  - 1차원의 시리즈 (Series)
  - 2차원의 데이터프레임 (DataFrame)

- 시리즈와 데이터프레임을 선언하고 데이터를 선택할 수 있음

- 데이터프레임 내 행/열을 추가, 삭제하고 원소값을 변경할 수 있음

- transpose, sort를 통해 데이터프레임을 조작할 수 있음

Thank you

junwonee@gmail.com