제조데이터 분석 및 시각화 ('22.02.22-'22.02.25) 3일차

Prepared by DaeKyeong Kim
Ph.D.

학습일정 및 내용

일차	시 간	교육 내용	세부 학습 내용	비고
1일차 (○○/○○)		Syllabus Python Basics		
	5교시		Syllabus와 작업 환경 이해	강의, 실습 및 코칭
	6교시		파이썬 기초	강의, 실습 및 코칭
	7교시		파이썬 기초	강의, 실습 및 코칭
	8교시		파일 읽고 쓰기	강의, 실습 및 코칭
2일차 (○○/○○)	1교시	Data Structures Data Cleanup		
	2교시			
	3교시			
	4교시			
	5교시		수치배열 데이터 패키지 소개	강의, 실습 및 코칭
	6교시		과학기술계산패키지 소개	강의, 실습 및 코칭
	7교시		sympy	강의, 실습 및 코칭
	8교시		데이터분석 패키지 소개	강의, 실습 및 코칭

학습일정 및 내용

일차	시 간	교육 내용	세부 학습 내용	비고
3일차 (○○/○○)	1교시	데이터 셋 적재하기와 Cleanup		
	2교시			
	3교시			
	4교시			
	5교시		데이터 셋 적재하기	강의, 실습 및 코칭
	6교시		Data Structures	강의, 실습 및 코칭
	7교시		Data Cleanup: Investigation, Matching, and Formatting	강의, 실습 및 코칭
	8교시		Data Cleanup: Investigation, Matching, and Formatting	강의, 실습 및 코칭
4일차 (○○/○○)	1교시	Data Structures Data Cleanup		
	2교시			
	3교시			
	4교시			
	5교시		시각화 패키지 소개	
	6교시		시각화 패키지 소개	
	7교시		시각화 패키지 소개	
	8교시		Iris를 통한 시각화 연습	

학습일정 및 내용

일차	시 간	교육 내용	세부 학습 내용	비고
5일차 (○○/○○)	1교시	제조데이터 분석 및 시각화 미니 프로젝트		
	2교시			
	3교시			
	4교시			
	5교시		미니 프로젝트	강의, 실습 및 코칭
	6교시		미니 프로젝트	강의, 실습 및 코칭
	7교시		미니 프로젝트	강의, 실습 및 코칭
	8교시		미니 프로젝트	강의, 실습 및 코칭

Contents

1. 데이터 셋 적재하기 6
2. Data Structures 51
3. Data Cleanup: Investigation, Matching, and Formatting 89

1

데이터 셋 적재하기

1. 샘플 데이터 셋 적재하기
2. 데이터 셋 적재하기

Unit 1

샘플 데이터 셋 적재하기

샘플 데이터셋 적재하기

❖ 데이터 분석 작업의 첫 번째 단계는 시스템으로 원본 데이터를 불러오는 것이다.

❖ Toy 데이터셋

✓ Boston house prices dataset

✓ 머신러닝과 여러 논문들에 활용이 많이 된 보스턴 하우스 가격 데이터셋

```
#Boston 데이터 셋 로드
from sklearn.datasets import load_boston
boston=load_boston()
boston.data.shape
```

- 데이터셋 요약

Samples total	506
Dimensionality	13
Features	real, positive
Targets	real 5. - 50.

샘플 데이터셋 적재하기

In [12]:
```
# 보스턴 데이터 셋 로드
from sklearn.datasets import load_boston

boston=load_boston()

boston.data.shape
```

Out [12]:
```
(506, 13)
```

In [13]:
```
boston_features=boston.data

boston_features[0]
```

Out [13]:
```
array([6.320e-03, 1.800e+01, 2.310e+00, 0.000e+00, 5.380e-01, 6.575e+00,
       6.520e+01, 4.090e+00, 1.000e+00, 2.960e+02, 1.530e+01, 3.969e+02,
       4.980e+00])
```

In [14]:
```
boston_features[0:4]
```

Out [14]:
```
array([[6.3200e-03, 1.8000e+01, 2.3100e+00, 0.0000e+00, 5.3800e-01,
        6.5750e+00, 6.5200e+01, 4.0900e+00, 1.0000e+00, 2.9600e+02,
        1.5300e+01, 3.9690e+02, 4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, 0.0000e+00, 4.6900e-01,
        6.4210e+00, 7.8900e+01, 4.9671e+00, 2.0000e+00, 2.4200e+02,
        1.7800e+01, 3.9690e+02, 9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, 0.0000e+00, 4.6900e-01,
        7.1850e+00, 6.1100e+01, 4.9671e+00, 2.0000e+00, 2.4200e+02,
        1.7800e+01, 3.9283e+02, 4.0300e+00],
       [3.2370e-02, 0.0000e+00, 2.1800e+00, 0.0000e+00, 4.5800e-01,
        6.9980e+00, 4.5800e+01, 6.0622e+00, 3.0000e+00, 2.2200e+02,
        1.8700e+01, 3.9463e+02, 2.9400e+00]])
```

샘플 데이터셋 적재하기

In [15]:
```
# head() 함수 이용해 값 조회
import pandas as pd
boston_df=pd.DataFrame(boston_features)

boston_df.head()
```

Out [15]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [16]:
```
boston_df.head(7)
```

Out [16]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43

샘플 데이터셋 적재하기

In [17]:
```
boston_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       506 non-null    float64
 1   1       506 non-null    float64
 2   2       506 non-null    float64
 3   3       506 non-null    float64
 4   4       506 non-null    float64
 5   5       506 non-null    float64
 6   6       506 non-null    float64
 7   7       506 non-null    float64
 8   8       506 non-null    float64
 9   9       506 non-null    float64
 10  10      506 non-null    float64
 11  11      506 non-null    float64
 12  12      506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
```

In [18]:
```
boston_df.columns
```

샘플 데이터셋 적재하기

- ❖ 머신러닝 작업의 첫 번째 단계는 시스템으로 원본 데이터를 불러오는 것이다.
- ❖ Toy 데이터셋
  - ✓ Diabetes dataset
    - ✓ 총 442명의 당뇨병 환자로부터 10가지 기준에 따른 연령, 성별 , 체질량 지수 등이 담겨진 데이터 셋

#Diabetes 데이터 셋 로드

```
from sklearn.datasets import load_diabetes
diabetes=load_diabetes()
diabetes.data.shape
```

- 데이터셋 요약

Samples total	442
Dimensionality	10
Features	real, -.2 < x < .2
Targets	integer 25 - 346

```
In [20]: #Diabetes 데이터 셋 로드

         from sklearn.datasets import load_diabetes
         diabetes=load_diabetes()
         diabetes.data.shape

Out[20]: (442, 10)
```

샘플 데이터셋 적재하기

- ❖ 머신러닝 작업의 첫 번째 단계는 시스템으로 원본 데이터를 불러오는 것이다.
- ❖ Toy 데이터셋
  - ✓ Optical recognition of handwritten digits dataset
    - ✓ 손으로 쓴 숫자의 이미지가 포함. 각 클래스는 10개의 숫자를 나타내는 클래스 데이터 셋

```
# Optical recognition of handwritten digits데이터 셋 로드

from sklearn.datasets import load_digits
digits=load_digits()
digits.data.shape

import matplotlib.pyplot as plt
plt.gray()
plt.matshow(digits.images[0])
plt.show()
```

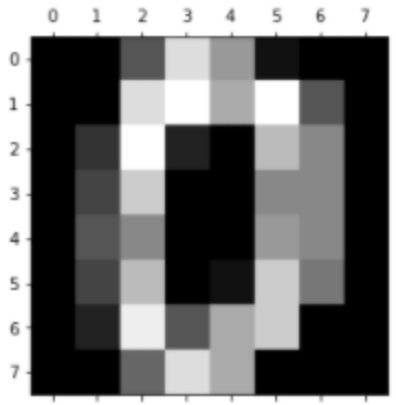
- 데이터셋 요약

Classes	10
Samples per class	~180
Samples total	1797
Dimensionality	64
Features	integers 0-16

샘플 데이터셋 적재하기

In [20]:
```
# Optical recognition of handwritten digits데이터 셋 로드

from sklearn.datasets import load_digits
digits=load_digits()
digits.data.shape

import matplotlib.pyplot as plt
plt.gray()
plt.matshow(digits.images[0])
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



In [21]:
```
digits.keys()
```

Out [21]:
```
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images', 'DESCR'])
```

In [22]:
```
# DESCR 키는 데이터셋에 대한 설명을 담고 있습니다.
digits['DESCR'][:70]
```

Out [22]:
```
'.. _digits_dataset:\n\nOptical recognition of handwritten digits dataset'
```

샘플 데이터셋 적재하기

- 머신러닝 작업의 첫 번째 단계는 시스템으로 원본 데이터를 불러오는 것이다.
- Toy 데이터셋
  - ✓ Wine recognition dataset
    - ✓ 이탈리아의 같은 지역 내의 3개의 다른 경작지에서 재배된 와인의 화학적 분석 결과. 3가지 종류의 와인에서 발견된 다른 성분들에 대해 13가지의 다른 측정치가 포함되어 있다.

```
from sklearn.datasets import load_wine
data=load_wine()
data.target[[10, 80, 140]]
list(data.target_names)
```

- 데이터셋 요약

Classes	3
Samples per class	[59,71,48]
Samples total	178
Dimensionality	13
Features	real, positive

```
In [31]: ##Wine데이터 셋 로드

         from sklearn.datasets import load_wine
         data=load_wine()
         data.target[[10, 80, 140]]

Out[31]: array([0, 1, 2])

In [32]: list(data.target_names)

Out[32]: ['class_0', 'class_1', 'class_2']
```

샘플 데이터셋 적재하기

❖ 머신러닝 작업의 첫 번째 단계는 시스템으로 원본 데이터를 불러오는 것이다.

❖ Toy 데이터셋

- ✓ Iris plants dataset
  - ✓ 아이리스 데이터셋

  - ✓ #iris 데이터 셋 로드
  - ✓ from sklearn.datasets import load_iris
  - ✓ iris_features=load_iris()

  - ✓ iris_features.data.shape

- 데이터셋 요약

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

샘플 데이터셋 적재하기

In [24]:
```
#iris 데이터 셋 로드
from sklearn.datasets import load_iris
iris_features=load_iris()

iris_features.data.shape
```

Out [24]:
```
(150, 4)
```

In [25]:
```
iris_features.keys()
```

Out [25]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

In [26]:
```
iris_features.target[[10,25,50]]
```

Out [26]:
```
array([0, 0, 1])
```

In [27]:
```
list(iris_features.target_names)
```

Out [27]:
```
['setosa', 'versicolor', 'virginica']
```

In [28]:
```
list(iris_features.feature_names)
```

Out [28]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [31]:
```
print(iris_features)
print(iris_features.DESCR)
print(iris_features.data)
print(iris_features.feature_names)
print(iris_features.target)
print(iris_features.target_names)
```

샘플 데이터셋 적재하기

In [45]:
```
import pandas as pd
import numpy as np
df = pd.DataFrame(data=iris_features.data, columns=iris_features.feature_names)
df['target'] = iris_features.target
# 0.0, 1.0, 2.0으로 표현된 label을 문자열로 매핑
df['target'] = df['target'].map({0:"setosa", 1:"versicolor", 2:"virginica"})
print(df)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  ₩
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

        target
0       setosa
1       setosa
2       setosa
3       setosa
4       setosa
..         ...
145  virginica
146  virginica
147  virginica
148  virginica
149  virginica

[150 rows x 5 columns]
```

사이킷런에 있는 데이터셋을 이용 모의 데이터셋 만들기

In [48]:
```
# 사이킷런에 있는 데이터셋을 이용 모의 데이터셋 만들기

#1. make_regression을 이용한 선형회귀에 사용할 데이터 셋 만들기
# 라이브러리를 임포트합니다.
from sklearn.datasets import make_regression

# 특성 행렬, 타깃 벡터, 정답 계수를 생성합니다.
features, target, coefficients = make_regression(n_samples = 100,
                                                 n_features = 3,
                                                 n_informative = 3,
                                                 n_targets = 1,
                                                 noise = 0.0,
                                                 coef = True,
                                                 random_state = 1)

# 특성 행렬과 타깃 벡터를 확인합니다.
print('특성 행렬\n', features[:3])
print('타깃 벡터\n', target[:3])
```

```
특성 행렬
 [[ 1.29322588 -0.61736206 -0.11044703]
 [-2.793085    0.36633201  1.93752881]
 [ 0.80186103 -0.18656977  0.0465673 ]]
타깃 벡터
 [-10.37865986  25.5124503   19.67705609]
```

사이킷런에 있는 데이터셋을 이용 모의 데이터셋 만들기

In [49]:
```
# 2. make_classification을 이용한 분류에 필요한 데이터셋 만들기

# 라이브러리를 임포트합니다.
from sklearn.datasets import make_classification

# 특성 행렬과 타깃 벡터를 생성합니다.
features, target = make_classification(n_samples = 100,
                                       n_features = 3,
                                       n_informative = 3,
                                       n_redundant = 0,
                                       n_classes = 2,
                                       weights = [.25, .75],
                                       random_state = 1)

# 특성 행렬과 타깃 벡터를 확인합니다.
print('특성 행렬₩n', features[:3])
print('타깃 벡터₩n', target[:3])
```

```
특성 행렬
 [[ 1.06354768 -1.42632219  1.02163151]
 [ 0.23156977  1.49535261  0.33251578]
 [ 0.15972951  0.83533515 -0.40869554]]
타깃 벡터
 [1 0 0]
```

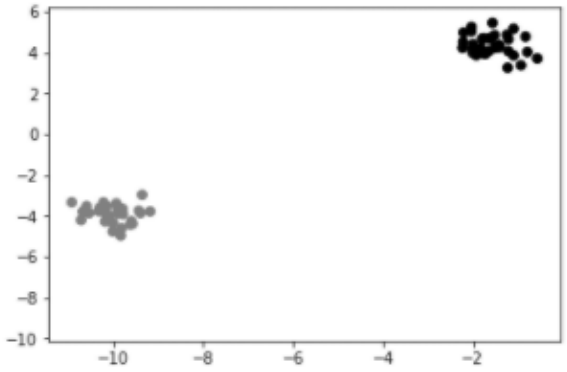
모의 데이터 셋 만들기

사이킷런에 있는 데이터셋을 이용 모의 데이터셋 만들기

In [50]:
```
# 3. make_blobs를 이용한 군집에 필요한 데이터 셋 만들기

# 라이브러리를 임포트합니다.
from sklearn.datasets import make_blobs

# 특성 행렬과 타깃 벡터를 생성합니다.
features, target = make_blobs(n_samples = 100,
                              n_features = 2,
                              centers = 3,
                              cluster_std = 0.5,
                              shuffle = True,
                              random_state = 1)

# 특성 행렬과 타깃 벡터를 확인합니다.
print('특성 행렬\n', features[:3])
print('타깃 벡터\n', target[:3])
```

```
특성 행렬
 [[ -1.22685609   3.25572052]
 [ -9.57463218  -4.38310652]
 [-10.71976941  -4.20558148]]
타깃 벡터
 [0 1 1]
```

In [51]:
```
# 4. make_blobs를 이용한 군집 데이터 셋 그림으로 확인

import matplotlib.pyplot as plt

plt.scatter(features[:,0], features[:,1], c=target)
plt.show()
```



Unit 2

데이터 셋 적재하기

Import CSV Data

❖Python의 pandas library의 read_csv() 함수를 사용해서 외부 text 파일, csv 파일을 불러와서 DataFrame으로 저장

❖불러오려는 text, csv 파일의 encoding 설정과 Python encoding 설정이 서로 맞지 않으면 UnicodeDecodeError 가 발생. 한글은 보통 'utf-8' 을 많이 사용하는데요, 만약 아래처럼 'utf-8' 코덱을 decode 할 수 없다고 에러 메시지가 나오는 경우가 있다.

```
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xc1 in position 26:
invalid start byte
```

❖ 실습을 위해 파일 업로드

Import CSV Data

In [52]:
```
import pandas as pd
url='diamonds.csv'
diamonds_df=pd.read_csv(url)
diamonds_df.head(5)
```

Out [52]:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [53]:
```
diamonds_df.tail(5)
```

Out [53]:

	carat	cut	color	clarity	depth	table	price	x	y	z
53935	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

Import CSV Data

In [54]:
```
print(diamonds_df)
```

```
       carat        cut color clarity  depth  table  price     x     y     z
0       0,23      Ideal     E     SI2   61,5   55,0    326  3,95  3,98  2,43
1       0,21    Premium     E     SI1   59,8   61,0    326  3,89  3,84  2,31
2       0,23       Good     E     VS1   56,9   65,0    327  4,05  4,07  2,31
3       0,29    Premium     I     VS2   62,4   58,0    334  4,20  4,23  2,63
4       0,31       Good     J     SI2   63,3   58,0    335  4,34  4,35  2,75
...      ...        ...   ...     ...    ...    ...    ...   ...   ...   ...
53935   0,72      Ideal     D     SI1   60,8   57,0   2757  5,75  5,76  3,50
53936   0,72       Good     D     SI1   63,1   55,0   2757  5,69  5,75  3,61
53937   0,70  Very Good     D     SI1   62,8   60,0   2757  5,66  5,68  3,56
53938   0,86    Premium     H     SI2   61,0   58,0   2757  6,15  6,12  3,74
53939   0,75      Ideal     D     SI2   62,2   55,0   2757  5,83  5,87  3,64

[53940 rows x 10 columns]
```

In [55]:
```
diamonds_11=pd.read_csv(url, skiprows=range(1,11), nrows=5)
diamonds_11
```

Out [55]:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.30	Good	J	SI1	64.0	55	339	4.25	4.28	2.73
1	0.23	Ideal	J	VS1	62.8	56	340	3.93	3.90	2.46
2	0.22	Premium	F	SI1	60.4	61	342	3.88	3.84	2.33
3	0.31	Ideal	J	SI2	62.2	54	344	4.35	4.37	2.71
4	0.20	Premium	E	SI2	60.2	62	345	3.79	3.75	2.27

Installing Python Packages

- ❖ xlrd에서 xlsx 확장자를 읽는 기능이 파이썬 3.9 버전 이상부터는 불안정해서 지원을 끊음
- ❖ 따라서, 기본적으로 제공하는 엔진은 xlrd인데 이 엔진이 지원이 중단되서 못쓰니 다른 엔진으로 교체를 해야 한다.

!pip install openpyxl

```
!pip install openpyxl

Requirement already satisfied: openpyxl in /usr/local/lib/python3.7/dist-packages (3.0.9)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.7/dist-packages (from openpyxl) (1.1.0)
```

Installing Python Packages

In [1]:
```
import pandas as pd
url='diamonds.xlsx'
diamonds_xl=pd.read_excel(url, engine = 'openpyxl')
print(diamonds_xl)
```

```
       carat        cut color clarity  depth  table  price     x     y     z
0       0.23      Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1       0.21    Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2       0.23       Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3       0.29    Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4       0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
...      ...        ...   ...     ...    ...    ...    ...   ...   ...   ...
53935   0.72      Ideal     D     SI1   60.8   57.0   2757  5.75  5.76  3.50
53936   0.72       Good     D     SI1   63.1   55.0   2757  5.69  5.75  3.61
53937   0.70  Very Good     D     SI1   62.8   60.0   2757  5.66  5.68  3.56
53938   0.86    Premium     H     SI2   61.0   58.0   2757  6.15  6.12  3.74
53939   0.75      Ideal     D     SI2   62.2   55.0   2757  5.83  5.87  3.64

[53940 rows x 10 columns]
```

In [2]:
```
diamonds_xl.head(5)
```

Out [2]:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

openpyxl 모듈 소개, 기본 사용법

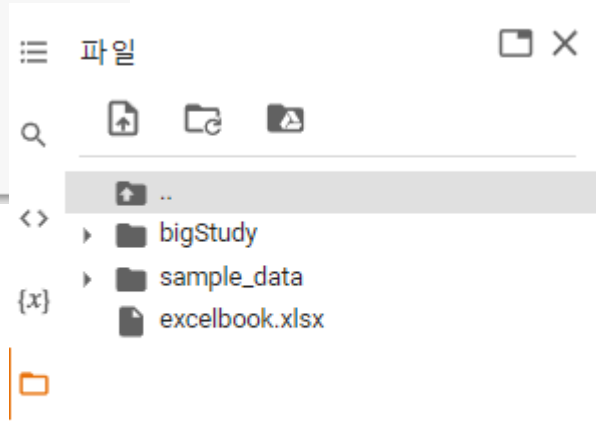
❖ openpyxl 모듈은 엑셀 파일 생성 및 조작이 가능한 모듈이다
- pip install openpyxl 을 실행하여 설치하고, import openpyxl 하여 사용한다

❖ openpyxl 모듈 사용시 주의할 사항은 다음과 같다
- 보안이 걸려 있는 파일을 열기 할 수 없고, “모두 새로 고침” 기능 지원 하지 않음
- 작업 속도가 XlsxWriter(대용량에 유리), PyExcelerate(문서화가 좋지 않음) 보다 느림
- 따라서, 작업에 따라 pywin32 및 pandas를 함께 사용하는 것이 필요하다

openpyxl 모듈 소개, 기본 사용법

❖ 빈 workbook 파일 만들고 저장하기

```
import openpyxl

dest_filename = 'excelbook.xlsx'
wb = openpyxl.Workbook()
sheet = wb.active
sheet.title = 'mysheet1'
sheet['A1'].value = 'name'
sheet['A2'].value = 'Hong Gil Dong'
sheet.cell(row=3, column=1).value = 20
wb.save(dest_filename)
```



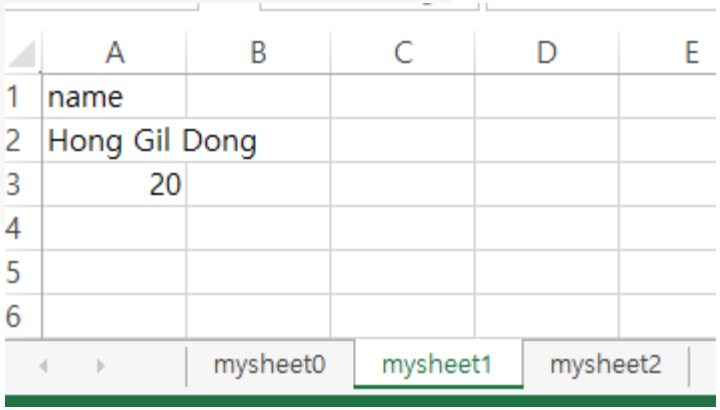
name
Hong Gil Dong
20

openpyxl 모듈 소개, 기본 사용법

❖ 엑셀 파일 열고 sheet 추가/특정 시트 정보 얻기

```
import openpyxl
target_filename = 'excelbook.xlsx'
dest_filename = 'excelbook_after.xlsx'
wb = openpyxl.load_workbook(target_filename)
print(wb.sheetnames)
wb.create_sheet(index=0, title='mysheet0')
wb.create_sheet(title='mysheet2')
print(wb.sheetnames)
snames = wb.sheetnames
sheet = wb[snames[1]]   # wb['mysheet1']
print(sheet['A1'].value)
print(sheet['A2'].value)
wb.save(dest_filename)
```

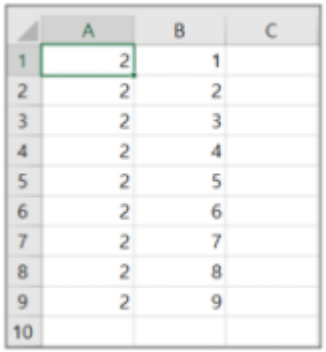
```
['mysheet1']
['mysheet0', 'mysheet1', 'mysheet2']
name
Hong Gil Dong
```



openpyxl 모듈 소개, 기본 사용법

❖ Range 사용하기

- range에서 cell에 접근하는 방법을 알아 보도록 한다
- range는 여러 셀을 포함하며, 아래와 같이 다양한 방법으로 지정할 수 있다
  - sheet['A1':'A9'] : A1에서 A9까지 범위 지정
  - sheet['B1:B9'] : B1부터 B9까지 범위 지정
  - sheet['1'] : 행번호 지정, 1행
  - sheet['A'] : 열번호 지정, A열
  - sheet['SheetName!A1:A5']: SheetName 시트의 A1에서 A5까지 범위 지정



	A	B	C
1	2	1	
2	2	2	
3	2	3	
4	2	4	
5	2	5	
6	2	6	
7	2	7	
8	2	8	
9	2	9	
10			

openpyxl 모듈 소개, 기본 사용법

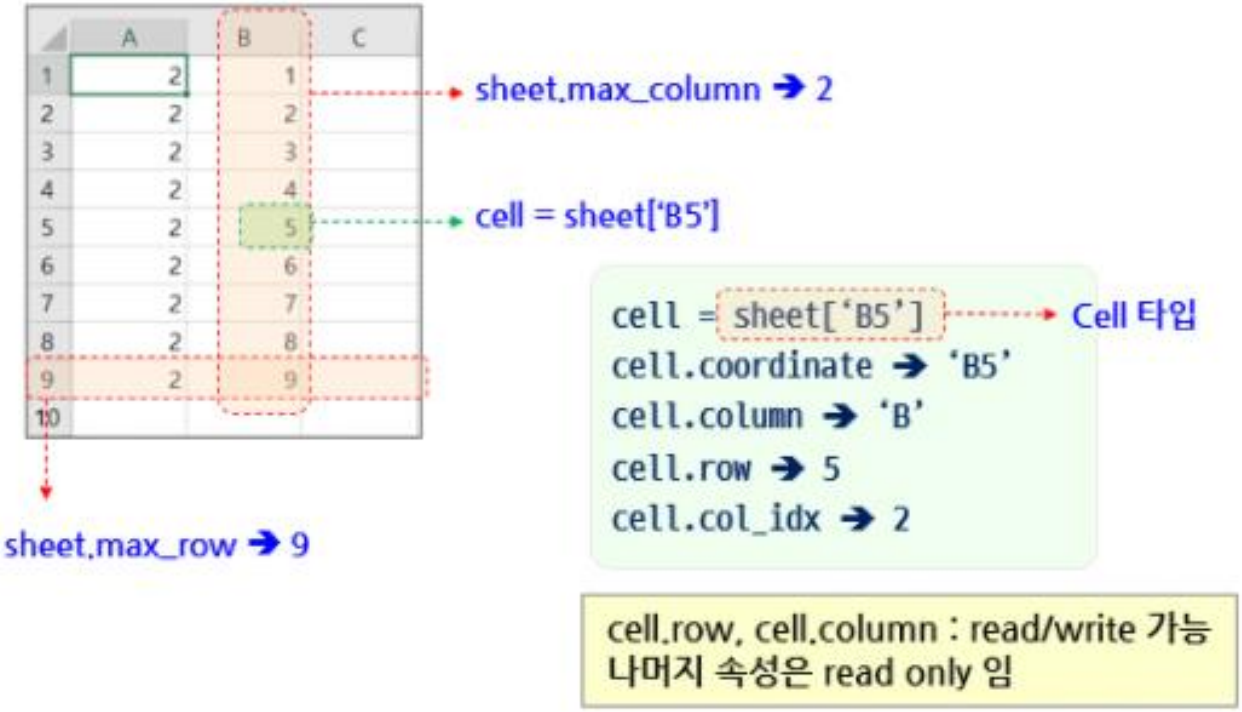
❖ Range 사용하기

```
import openpyxl

wb = openpyxl.Workbook()
sheet = wb.active
a = sheet['A1':'B9']   # ((한 개의 행에 대한 셀의 모음), (), ())

for no, xy in enumerate(a, start=1):
    x, y = xy
    x.value = 2
    y.value = no

for x, y in a:
    print(f'{x.value} * {y.value} = {x.value*y.value}')

for x in sheet['1']:
    print(x, x.value)

a = sheet['A1']
b = sheet['1']
print(a, b)
wb.save('gugu.xlsx')
```

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
<Cell 'Sheet'.A1> 2
<Cell 'Sheet'.B1> 1
<Cell 'Sheet'.A1> (<Cell 'Sheet'.A1>, <Cell 'Sheet'.B1>)
```

openpyxl 모듈 소개, 기본 사용법

❖ 시트와 셀의 정보 얻기



openpyxl 모듈 소개, 기본 사용법

❖ 시트와 셀의 정보 얻기

```
import openpyxl

wb = openpyxl.load_workbook('gugu.xlsx')
sheet = wb['Sheet']
print(sheet.max_row)
print(sheet.max_column)
cell = sheet['B5']
print(type(cell))
print(cell.coordinate)
print(cell.column)
print(cell.row)
print(cell.col_idx)
cell.row = 10
print(cell.coordinate)
#cell.coordinate = 'A1'
#print(cell.coordinate)
```

```
9
2
<class 'openpyxl.cell.cell.Cell'>
B5
2
5
2
B10
```

openpyxl 모듈 소개, 기본 사용법

❖ 수식 작성하기

- 등호로 시작하는 수식을 작성한다
- cell에는 수식이 적혀 있고, 눈에 보일 때는 수식의 결과가 표시된다

```
import openpyxl

wb = openpyxl.load_workbook('gugu.xlsx')
sheet = wb['Sheet']   # wb[wb.sheetnames[0]], wb.active
for x in sheet['C1:C9']:
    #print(x[0])
    cell = x[0]
    cell.value = f'=A{cell.row}*B{cell.row}'

for x in sheet['C1:C9']:
    print(x[0].value)

wb.save('gugu_2dan.xlsx')
```

```
=A1*B1
=A2*B2
=A3*B3
=A4*B4
=A5*B5
=A6*B6
=A7*B7
=A8*B8
=A9*B9
```

	A	B	C
1	2	1	2
2	2	2	4
3	2	3	6
4	2	4	8
5	2	5	10
6	2	6	12
7	2	7	14
8	2	8	16
9	2	9	18

openpyxl sheet 구조 조작

❖ sheet 구조 조작 메서드

함수	예시
sheet.delete_rows(행번호, [연속행의 수]) sheet.delete_cols(열번호, [연속열의 수]) sheet.insert_rows(행번호, [연속행의 수]) sheet.insert_cols(열번호, [연속열의 수])	▪ sheet.delete_rows(3) # 3 행 삭제 ▪ sheet.delete_rows(6,3) # 6:8 행 삭제 ▪ sheet.delete_cols(3) # C 열 삭제 ▪ sheet.delete_cols(6,3) # F:H 열 삭제 ➔ 삽입도 동일한 방식으로 동작한다
sheet.move_range(범위, cols=정수, rows=정수)	▪ sheet.move_range('C1:F4', cols=-1, rows=3) ➔ [C1:F4]의 내용이 [B4:D7]로 이동된다
sheet.merge_cells(범위)	▪ sheet.merge_cells('A1:B5') ➔ 가장 왼쪽 상단의 내용만 유지된다
sheet.unmerge_cells(범위)	▪ sheet.unmerge_cells('A1:B5') ➔ 병합 해지가 불가능하면 Error 발생
sheet.merged_cells.ranges	▪ 병합된 영역에 대한 객체(MergeCell)의 list ▪ MergeCell의 str()을 구하면 영역문자열이 반환됨 예) 'A1:A5'

openpyxl sheet 구조 조작

❖ 열 삭제 실습

❖ simpleTable.xlsx 다음 시트에서 B, E, F, G 열을 삭제

A	B	C	D	E	F	G	H	I
1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

A	B	C	D	E	F
1	3	4	8	9	
1	3	4	8	9	
1	3	4	8	9	
1	3	4	8	9	

openpyxl sheet 구조 조작

❖ 열 삭제 실습

❖ simpleTable.xlsx 다음 시트에서 B, E, F, G 열을 삭제

```
# 삭제/삽입 수행 후에는 행/열 번호가 변경 되기 때문에 주의 해야 한다
# B 열을 삭제하면, 뒤에 있던 C 열이 B열이 된다
# B,E,F,G를 삭제하기 위한 방법은 다음과 같다
# 2번 삭제 후, 4번부터 3개 열을 삭제하거나, 2번 삭제 후, 4번을 세 번 삭제 한다

import openpyxl
rfilename = 'bigStudy/dataset/simpleTable.xlsx'
wfilename = 'simpleTable_after.xlsx'
wb = openpyxl.load_workbook(rfilename)
sheet = wb.active
sheet.delete_cols(2)
sheet.delete_cols(4,3)
wb.save(wfilename)
print('simpleTable_after.xlsx 저장 완료')
```

```
simpleTable_after.xlsx 저장 완료
```

openpyxl sheet 구조 조작

❖ 병합 해제하기

A	B	C
주소	이름	연락처
서울	홍길동	010-111-1111
		111-1111
	박문수	010-111-1112
		111-1112
광주	황진희	010-111-1114
		111-1114
대구	이순신	010-111-1115

A	B	C
주소	이름	연락처
서울	홍길동	010-111-1111
서울	홍길동	111-1111
서울	박문수	010-111-1112
서울	박문수	111-1112
광주	황진희	010-111-1114
광주	황진희	111-1114
대구	이순신	010-111-1115

openpyxl sheet 구조 조작

❖ 병합 해제하기

```
import openpyxl
import copy

rfilename = 'bigStudy/dataset/mergedTable.xlsx'
wfilename = 'mergedTable_after.xlsx'

def copy_value(rng, sheet):
    value = sheet[rng][0][0].value
    for row in sheet[rng]:
        for col in row:
            col.value = value

wb = openpyxl.load_workbook(rfilename)
sheet = wb.active
mcells = copy.copy(sheet.merged_cells.ranges)
for rng in mcells:
    print(type(rng), str(rng))
    rng = str(rng)
    sheet.unmerge_cells(rng)
    copy_value(rng, sheet)
wb.save(wfilename)
```

```
<class 'openpyxl.worksheet.merge.MergedCellRange'> A2:A5
<class 'openpyxl.worksheet.merge.MergedCellRange'> B2:B3
<class 'openpyxl.worksheet.merge.MergedCellRange'> B4:B5
<class 'openpyxl.worksheet.merge.MergedCellRange'> A6:A7
<class 'openpyxl.worksheet.merge.MergedCellRange'> B6:B7
```

openpyxl sheet 구조 조작

❖ 병합 해제하기

```
import pandas as pd

df = pd.read_excel('bigStudy/dataset/mergedTable.xlsx')
df = df.fillna(method='ffill')
print(df)
df.to_excel('mergedTable_after2.xlsx', index=False)
```

```
   주소   이름          연락처
0  서울  홍길동  010-111-1111
1  서울  홍길동      111-1111
2  서울  박문수  010-111-1112
3  서울  박문수      111-1112
4  광주  황진희  010-111-1114
5  광주  황진희      111-1114
6  대구  이순신  010-111-1115
```

Import JSON Data

❖ JSON파일 내보내기

In [3]:
```
import csv
import json

csvfile=open('diamonds.csv','r')
jsonfile=open('diamonds.json','w')

fieldnames=("FirstName","LastName","IDNumber","Message")
reader=csv.DictReader(csvfile, fieldnames)

out=json.dumps([row for row in reader])
jsonfile.write(out)
```

Out [3]: 7303173

Import JSON Data

❖ JSON파일 적재하기

In [4]:
```
import pandas as pd

url='diamonds.json'

diamonds_json=pd.read_json(url, orient='columns')

diamonds_json.head(5)
```

Out [4]:

	FirstName	LastName	IDNumber	Message	null
0	carat	cut	color	clarity	[depth, table, price, x, y, z]
1	0.23	Ideal	E	SI2	[61.5, 55, 326, 3.95, 3.98, 2.43]
2	0.21	Premium	E	SI1	[59.8, 61, 326, 3.89, 3.84, 2.31]
3	0.23	Good	E	VS1	[56.9, 65, 327, 4.05, 4.07, 2.31]
4	0.29	Premium	I	VS2	[62.4, 58, 334, 4.2, 4.23, 2.63]

Import XML Data

```
from xml.etree import ElementTree as ET
tree = ET.parse('data-text.xml')
root = tree.getroot()
print dir(root)
```

Import XML Data

In [20]:
```
from xml.etree import ElementTree as ET
tree = ET.parse('Data.xml')
root = tree.getroot()
root.tag
```

Out [20]:
```
'data'
```

In [21]:
```
root.attrib
```

Out [21]:
```
{}
```

In [22]:
```
for child in root:
    print(child.tag, child.attrib)
```

```
country {'name': 'Liechtenstein'}
country {'name': 'Singapore'}
country {'name': 'Panama'}
```

SQLite 사용

❖ Siri를 사용하여 휴대폰에서 번호를 찾아 본 적이 있습니까? 검색해 본 적이 있습니까?구글? Twitter 또는 Instagram에서 해시 태그를 클릭 한 적이 있습니까? 이들 각각작업에는 간단한 검색과 데이터베이스 (또는 일련의 데이터베이스)의 응답이 포함됩니다.

In [5]:
```
import sqlite3
print(sqlite3.version)
print(sqlite3.sqlite_version)
```

```
2.6.0
3.22.0
```

In [6]:
```
# DB 연결, 커서 획득
# DB 생성 (오토 커밋)
conn = sqlite3.connect("test.db", isolation_level=None)
# 커서 획득
c = conn.cursor()
# 테이블 생성 (데이터 타입은 TEXT, NUMERIC, INTEGER, REAL, BLOB 등)
c.execute("CREATE TABLE IF NOT EXISTS table1 ₩
    (id integer PRIMARY KEY, name text, birthday text)")
```

Out [6]:
```
<sqlite3.Cursor at 0x7fa105b2d960>
```

SQLite 사용

In [7]:
```
# 데이터 삽입 방법 1
c.execute("INSERT INTO table1 ₩
    VALUES(1, 'LEE', '1987-00-00')")
```

Out [7]:
```
<sqlite3.Cursor at 0x7fa105b2d960>
```

In [8]:
```
# 데이터 삽입 방법 2
c.execute("INSERT INTO table1(id, name, birthday) ₩
    VALUES(?,?,?)", ₩
    (2, 'KIM', '1990-00-00'))
```

Out [8]:
```
<sqlite3.Cursor at 0x7fa105b2d960>
```

In [9]:
```
test_tuple = (
    (3, 'PARK', '1991-00-00'),
    (4, 'CHOI', '1999-00-00'),
    (5, 'JUNG', '1989-00-00')
)
c.executemany("INSERT INTO table1(id, name, birthday) VALUES(?,?,?)", test_tuple)
```

Out [9]:
```
<sqlite3.Cursor at 0x7fa105b2d960>
```

In [10]:
```
# 데이터 불러오기
c.execute("SELECT * FROM table1")
print(c.fetchone())
print(c.fetchone())
print(c.fetchall())
```

```
(1, 'LEE', '1987-00-00')
(2, 'KIM', '1990-00-00')
[(3, 'PARK', '1991-00-00'), (4, 'CHOI', '1999-00-00'), (5, 'JUNG', '1989-00-00')]
```

SQLite 사용

In [11]:
```
# 전체 데이터
c.execute("SELECT * FROM table1")
print(c.fetchall())
```

```
[(1, 'LEE', '1987-00-00'), (2, 'KIM', '1990-00-00'), (3, 'PARK', '1991-00-00'), (4, 'CHOI', '1999-00-00'), (5, 'JUNG', '1989-00-0
0')]
```

In [12]:
```
# 리스트 형태로 출력
# 방법 1
c.execute("SELECT * FROM table1")
for row in c.fetchall():
    print(row)
# 방법 2
for row in c.execute("SELECT * FROM table1 ORDER BY id ASC"):
    print(row)
```

```
(1, 'LEE', '1987-00-00')
(2, 'KIM', '1990-00-00')
(3, 'PARK', '1991-00-00')
(4, 'CHOI', '1999-00-00')
(5, 'JUNG', '1989-00-00')
(1, 'LEE', '1987-00-00')
(2, 'KIM', '1990-00-00')
(3, 'PARK', '1991-00-00')
(4, 'CHOI', '1999-00-00')
(5, 'JUNG', '1989-00-00')
```

SQLite 사용

In [13]:
```
# 데이터 수정

# 방법 1
c.execute("UPDATE table1 SET name=? WHERE id=?", ('NEW1', 1))
# 방법 2
c.execute("UPDATE table1 SET name=:name WHERE id=:id", {"name": 'NEW2', 'id': 3})
# 방법 3
c.execute("UPDATE table1 SET name='%s' WHERE id='%s'" % ('NEW3', 5))
# 확인
for row in c.execute('SELECT * FROM table1'):
    print(row)
```

```
(1, 'NEW1', '1987-00-00')
(2, 'KIM', '1990-00-00')
(3, 'NEW2', '1991-00-00')
(4, 'CHOI', '1999-00-00')
(5, 'NEW3', '1989-00-00')
```

In [14]:
```
# 데이터 삭제하기

# 방법 1
c.execute("DELETE FROM table1 WHERE id=?", (1,))
# 방법 2
c.execute("DELETE FROM table1 WHERE id=:id", {'id': 3})
# 방법 3
c.execute("DELETE FROM table1 WHERE id='%s'" % 5)
# 확인
for row in c.execute('SELECT * FROM table1'):
    print(row)
```

```
(2, 'KIM', '1990-00-00')
(4, 'CHOI', '1999-00-00')
```

SQLite 사용

In [15]:
```
# DB 백업하기 (dump)

with conn:
    with open('dump.sql', 'w') as f:
        for line in conn.iterdump():
            f.write('%s\n' % line)
        print('Completed.')
```

```
Completed.
```

In [16]:
```
# DB 연결 해제
conn.close()
```

3

Data Structures

1. Series
2. DataFrame

학습목표

- 수집, 저장, 데이터 셋 적재 후 내용을 DataFrame에 담을 수 있다.
- DataFrame을 사용할 수 있다.

Unit 1

Series

PANDAS

- 수집, 저장, 데이터 셋 적재 Gathering Data 후 DataFrame에 담을 수 있어야 한다. 대부분의 데이터는 시계열(series)이나 표(table)의 형태로 나타낼 수 있다.
- 데이터 랭글링에 사용되는 가장 일반적인 데이터 구조는 데이터프레임이다. 사용하기 쉽고 기능이 많음. 판다스(Pandas) 패키지는 이러한 데이터를 다루기 위한 시리즈(Series) 클래스와 데이터프레임(DataFrame) 클래스를 제공한다.
- 데이터프레임은 표 형식 데이터로서 PANDAS를 통해 사용할 수 있다.
- 판다스 패키지를 사용하기 위해 우선 임포트를 해야 한다. 판다스 패키지는 pd라는 별칭으로 임포트하는 것이 관례이므로 여기에서도 해당 관례를 따르도록 한다.

```
import pandas as pd
```

PANDAS

❖ PANDAS를 이해하기위한 핵심 중 하나는 데이터 모델을 이해하는 것입니다. Pandas의 핵심에는 세 가지 데이터 구조가 있습니다.

DATA STRUCTURE	DIMENSIONALITY	SPREADSHEET ANALOG
Series	1D	Column
DataFrame	2D	Single Sheet
Panel	3D	Multiple Sheets

❖ 가장 널리 사용되는 데이터 구조는 각각 배열 데이터와 테이블 형식 데이터를 처리하는 Series 및 DataFrame. 스프레드 시트 세계와의 비유는 이러한 유형 간의 기본적인 차이점을 보여준다. DataFrame은 행과 열이 있는 시트와 비슷하지만 Series는 데이터의 단일 열과 비슷합니다. 패널은 시트 그룹. 마찬가지로, 팬더에서 패널은 여러 데이터 프레임을 가질 수 있으며, 각 프레임은 차례로 여러 시리즈를 가질 수 있다.

개념

- 시리즈는 Python의 목록과 유사한 1 차원 데이터를 모델링하는 데 사용. Series 객체에는 색인 및 이름을 포함하여 몇 비트 더 많은 데이터가 있다. 팬더를 통한 일반적인 아이디어는 축의 개념입니다. 계열은 1 차원이므로 단일 축인 인덱스를 갖는다.
- 특징
- 레이블 또는 데이터의 위치를 지정한 추출 가능.
- 산술 연산 가능.

Artist	Data
0	145
1	142
2	38
3	13

pandas Series 예

```
>>> import pandas as pd
>>> songs2 = pd.Series([145, 142, 38, 13])
>>> songs2
0 145
1 142
2 38
3 13
Name: counts, dtype: int64
```

```
In [3]: import pandas as pd
        songs1 = pd.Series([145, 142, 38, 13])
        songs1

Out[3]: 0    145
        1    142
        2     38
        3     13
        dtype: int64
```

Similar to NumPy

```
In [1]: import numpy as np
        numpy_ser = np.array([145, 142, 38, 13])

In [4]: songs1[1]
Out[4]: 142

In [5]: numpy_ser[1]
Out[5]: 142

In [6]: songs1.mean()
Out[6]: 84.5

In [7]: numpy_ser.mean()
Out[7]: 84.5
```

Series CRUD

Creation

색인은 목록을 사용하여 두 번째 매개 변수로 지정.

```
>>> george_dupe = pd.Series([10, 7, 1, 22],index=['1968', '1969', '1970', '1970'],name='George Songs')
>>> george_dupe
1968 10
1969 7
1970 1
1970 22
Name: George Songs, dtype: int64
```

```
In [9]: george_dupe

Out[9]: 1968    10
        1969     7
        1970     1
        1970    22
        Name: George Songs, dtype: int64
```

Reading

```
>>> george_dupe['1968']
10
```

```
In [22]: george_dupe['1968']
Out[22]: 10

In [27]: george_dupe != 2
Out[27]: 1968    True
         1969    True
         1970    True
         1971    True
         Name: George Songs, dtype: bool

In [28]: george_dupe.loc[george_dupe != 2]
Out[28]: 1968    10
         1969     7
         1970     1
         1971    22
         Name: George Songs, dtype: int64
```

Series CRUD

Reading

```
In [14]: george_dupe != 2

Out[14]: 1968    True
         1969    True
         1970    True
         1970    True
         Name: George Songs, dtype: bool

In [15]: george_dupe.loc[george_dupe != 2]

Out[15]: 1968    10
         1969     7
         1970     1
         1970    22
         Name: George Songs, dtype: int64
```

Updating

```
>>> george_dupe['1969'] = 6
>>> george_dupe['1969']
6
```

```
In [17]: george_dupe['1969'] = 6
         george_dupe['1969']

Out[17]: 6
```

Deletion

```
>>> del george_dupe['1971']
>>> george_dupe
1968 10
1969 6
1970 1
Name: George Songs, dtype: int64
```

```
In [30]: del george_dupe['1971']
         george_dupe

Out[30]: 1968    10
         1969     6
         1970     1
         Name: George Songs, dtype: int64
```

Indexing Summary

METHOD	WHEN TO USE
속성 액세스	이름이 유효한 경우 단일 색인 이름에 대한 값 가져 오기속성 이름.
인덱스액세스	이름이 아닌 경우 단일 인덱스 이름에 대한 값 가져 오기 / 설정유효한 속성 이름.
.iloc	인덱스 위치 또는 위치별로 값 가져 오기 / 설정. (반 개방슬라이스 간격)
.loc	색인 레이블로 값 가져 오기 / 설정. (슬라이스 폐쇄 간격)
.iat	인덱스 위치별로 numpy 배열 결과 가져 오기 / 설정.
.at	인덱스 레이블로 numpy 배열 결과 가져 오기 / 설정

Series Indexing

.iloc and .loc

```
In [24]: george_dupe.iloc[0]
Out[24]: 10

In [23]: george_dupe.loc["1968"]
Out[23]: 10

In [26]: george_dupe.iloc[0:3]
Out[26]: 1968    10
         1969     7
         1970     1
         Name: George Songs, dtype: int64

In [25]: george_dupe.loc["1968":"1970"]
Out[25]: 1968    10
         1969     7
         1970     1
         Name: George Songs, dtype: int64
```

Series Indexing

.at and .iat

>>> george_dupe = pd.Series([10, 7, 1, 22], index=['1968', '1969', '1970', '1970'], name='George Songs')

```
In [31]: george_dupe = pd.Series([10, 7, 1, 22], index=['1968', '1969', '1970', '1970'], name='George Songs'

In [32]: george_dupe.at['1970']

Out[32]: 1970     1
         1970    22
         Name: George Songs, dtype: int64

In [33]: george_dupe.loc['1970']

Out[33]: 1970     1
         1970    22
         Name: George Songs, dtype: int64
```

Slicing

:

SLICE	RESULT
0:1	First item
:1	First item (start default is 0)
:-2	처음부터 두 번째 항목부터 마지막 항목까지
::2	다른 모든 항목을 건너 뛰고 처음부터 끝까지 가져옵니다.

```
In [36]: george_dupe.iloc[0:2]
Out[36]: 1968    10
         1969     7
         Name: George Songs, dtype: int64
```

Boolean Arrays

...

```
>>> mask = george > 7
>>> mask
1968 True
1969 False
Name: George Songs, dtype: bool
```

```
In [37]: mask = george_dupe > 7
         mask

Out[37]: 1968     True
         1969    False
         1970    False
         1970     True
         Name: George Songs, dtype: bool
```

Iteration

```
>>> songs_66 = pd.Series([3, None , 11, 9], index=['George', 'Ringo', 'John', 'Paul'], name='Counts')
>>> songs_69 = pd.Series([18, 22, 7, 5], index=[ 'John', 'Paul','George', 'Ringo'], name='Counts')
```

```
In [38]: songs_66 = pd.Series([3, None , 11, 9], index=['George', 'Ringo', 'John', 'Paul'], name='Counts')

In [41]: songs_69 = pd.Series([18, 22, 7, 5], index=[ 'John', 'Paul','George', 'Ringo'], name='Counts')

In [42]: for value in songs_66:
             print(value)

3.0
nan
11.0
9.0
```

Overloaded operations

```
>>> songs_66 + songs_69
George 10.0
John 29.0
Paul 31.0
Ringo NaN
Name: Counts, dtype: float64
```

```
In [43]: songs_66 + songs_69
Out[43]: George    10.0
         John      29.0
         Paul      31.0
         Ringo      NaN
         Name: Counts, dtype: float64
```

Reset Index

- 색인을 재설정하는 몇 가지 방법
- 선택, 플로팅, 결합 및 기타 방법을 결정할 수 있기 때문에 인덱스에 따라 종종 인덱스 값을 변경하는 것이 유용.

```
In [47]: songs_66.reset_index()
```

Out[47]:

	index	Counts
0	George	3.0
1	Ringo	NaN
2	John	11.0
3	Paul	9.0

Statistics

songs_66 = pd.Series([3, None , 11, 9], index=['George', 'Ringo', 'John', 'Paul'], name='Counts')

songs_66

```
In [49]: songs_66 = pd.Series([3, None , 11, 9], index=['George', 'Ringo', 'John', 'Paul'], name='Counts')

         songs_66

Out[49]: George     3.0
         Ringo      NaN
         John      11.0
         Paul       9.0
         Name: Counts, dtype: float64

In [50]: songs_66.sum(0)

Out[50]: 23.0

In [51]: songs_66.mean()

Out[51]: 7.666666666666667

In [52]: songs_66.median()

Out[52]: 9.0
```

Statistics

```
In [53]: songs_66.quantile()
Out[53]: 9.0

In [55]: songs_66.quantile(.1)
Out[55]: 4.2

In [56]: songs_66.quantile(.9)
Out[56]: 10.6

In [57]: songs_66.describe()
Out[57]: count     3.000000
         mean      7.666667
         std       4.163332
         min       3.000000
         25%       6.000000
         50%       9.000000
         75%      10.000000
         max      11.000000
         Name: Counts, dtype: float64
```

Statistics

```
In [58]: songs_66.min()
Out[58]: 3.0

In [59]: songs_66.idxmin()
Out[59]: 'George'

In [61]: songs_66.max()
Out[61]: 11.0

In [62]: songs_66.idxmax()
Out[62]: 'John'

In [63]: songs_66.var()
Out[63]: 17.333333333333336

In [64]: songs_66.std()
Out[64]: 4.163331998932266

In [65]: songs_66.mad()
Out[65]: 3.1111111111111107
```

Unit 2

DataFrame

개요

- DataFrame은 행과 열에 레이블을 가진 2차원 데이터이다.
- 열마다 다른 형태를 가질 수 있음
- 테이블형 데이터에 대해 불러오기, 데이터 쓰기가 가능
- DataFrame끼리 여러 가지 조건을 사용한 결합 처리가 가능
- 크로스 집계가 가능

데이터 프레임 만들기

❖ 데이터 프레임 생성 후 개별적으로 각 열 정의

In [1]:
```
# 라이브러리를 임포트합니다.
import pandas as pd

# 데이터프레임을 만듭니다.
dataframe = pd.DataFrame()

# 열을 추가합니다.
dataframe['Name'] = ['Jacky Jackson', 'Steven Stevenson']
dataframe['Age'] = [38, 25]
dataframe['Driver'] = [True, False]

# 데이터프레임을 확인합니다.
dataframe
```

Out[1]:

	Name	Age	Driver
0	Jacky Jackson	38	True
1	Steven Stevenson	25	False

데이터 프레임 만들기

❖ 데이터프레임 객체를 만든 후 새로운 행 추가

In [2]:
```
# 열을 만듭니다.
new_person = pd.Series(['Molly Mooney', 40, True], index=['Name','Age','Driver'])

# 열을 추가합니다.
dataframe.append(new_person, ignore_index=True)
```

Out[2]:

	Name	Age	Driver
0	Jacky Jackson	38	True
1	Steven Stevenson	25	False
2	Molly Mooney	40	True

데이터 프레임 만들기

❖ 넘파이 배열을 통해, 열 이름을 컬럼 매개변수에 지정해 만듬

In [3]:
```
import numpy as np
import pandas as pd

data = [ ['Jacky Jackson', 38, True], ['Steven Stevenson', 25, False] ]

matrix = np.array(data)
pd.DataFrame(matrix, columns=['Name', 'Age', 'Driver'])
```

Out[3]:

	Name	Age	Driver
0	Jacky Jackson	38	True
1	Steven Stevenson	25	False

❖ 원본 리스트 전달로 생성

In [4]:
```
pd.DataFrame(data, columns=['Name', 'Age', 'Driver'])
```

Out[4]:

	Name	Age	Driver
0	Jacky Jackson	38	True
1	Steven Stevenson	25	False

데이터 프레임 만들기

❖ 넘파이 배열을 통해, 열 이름을 컬럼 매개변수에 지정해 만듬

In [3]:
```
import numpy as np
import pandas as pd

data = [ ['Jacky Jackson', 38, True], ['Steven Stevenson', 25, False] ]

matrix = np.array(data)
pd.DataFrame(matrix, columns=['Name', 'Age', 'Driver'])
```

Out[3]:

	Name	Age	Driver
0	Jacky Jackson	38	True
1	Steven Stevenson	25	False

❖ 원본 리스트 전달로 생성

In [4]:
```
pd.DataFrame(data, columns=['Name', 'Age', 'Driver'])
```

Out[4]:

	Name	Age	Driver
0	Jacky Jackson	38	True
1	Steven Stevenson	25	False

데이터 프레임 만들기

❖ 열 이름과 매핑한 딕셔너리 사용 생성

```
In [5]: data = {'Name': ['Jacky Jackson', 'Steven Stevenson'],
                'Age': [38, 25],
                'Driver': [True, False]}
        pd.DataFrame(data)
```

Out[5]:

	Name	Age	Driver
0	Jacky Jackson	38	True
1	Steven Stevenson	25	False

❖ 열과 값을 매핑한 딕셔너리를 리스트로 전달해 생성

```
In [6]: data = [ {'Name': 'Jacky Jackson', 'Age': 38, 'Driver': True},
                 {'Name': 'Steven Stevenson', 'Age': 25, 'Driver': False} ]
        pd.DataFrame(data, index=['row1', 'row2'])
```

Out[6]:

	Name	Age	Driver
row1	Jacky Jackson	38	True
row2	Steven Stevenson	25	False

데이터 프레임 적재 및 확인

❖ 데이터 적재

In [16]:
```
# 라이브러리를 임포트합니다.
import pandas as pd

# 데이터를 적재합니다.
dataframe = pd.read_csv("bigStudy/titanic.csv")

# 두 개의 행을 확인합니다.
dataframe.head(2)
```

Out [16]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

❖ 차원 확인

In [17]:
```
# 차원을 확인합니다.
dataframe.shape
```

Out [17]: (891, 12)

데이터 프레임 적재 및 확인

❖ 기술 통계값 요약

In [18]:
```
# 통계값을 확인합니다.
dataframe.describe()
```

Out [18]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

최솟값, 최댓값, 합, 평균 계산 및 개수 세기

❖ 기술 통계값 요약

In [15]:
```
# 통계값을 계산합니다.
print('최댓값:', dataframe['Age'].max())
print('최솟값:', dataframe['Age'].min())
print('평균:', dataframe['Age'].mean())
print('합:', dataframe['Age'].sum())
print('카운트:', dataframe['Age'].count())
```

```
최댓값: 80.0
최솟값: 0.42
평균: 29.69911764705882
합: 21205.17
카운트: 714
```

In [16]:
```
# 카운트를 출력합니다.
dataframe.count()
```

Out [16]:
```
PassengerId    891
Survived       891
Pclass         891
Name           891
Gender         891
Age            714
SibSp          891
Parch          891
Ticket         891
Fare           891
Cabin          204
Embarked       889
dtype: int64
```

최솟값, 최댓값, 합, 평균 계산 및 개수 세기

❖ 공분산과 상관계수

In [17]:
```
# 수치형 열의 공분산을 계산합니다.
dataframe.cov()
```

Out [17]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	66231.000000	-0.626966	-7.561798	138.696504	-16.325843	-0.342697	161.883369
Survived	-0.626966	0.236772	-0.137703	-0.551296	-0.018954	0.032017	6.221787
Pclass	-7.561798	-0.137703	0.699015	-4.496004	0.076599	0.012429	-22.830196
Age	138.696504	-0.551296	-4.496004	211.019125	-4.163334	-2.344191	73.849030
SibSp	-16.325843	-0.018954	0.076599	-4.163334	1.216043	0.368739	8.748734
Parch	-0.342697	0.032017	0.012429	-2.344191	0.368739	0.649728	8.661052
Fare	161.883369	6.221787	-22.830196	73.849030	8.748734	8.661052	2469.436846

In [18]:
```
# 수치형 열의 상관계수를 계산합니다.
dataframe.corr()
```

Out [18]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

데이터 선택하기

❖ 레이블 사용과 모든 행(열)을 지정하는 경우

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

df = pd.DataFrame(
    [[1, 10, 100], [2, 20, 200], [3, 30, 300]],
    index=["r1", "r2", "r3"],
    columns=["c1", "c2", "c3"],
)
df
```

Out [1]:

	c1	c2	c3
r1	1	10	100
r2	2	20	200
r3	3	30	300

In [2]:
```
df.loc["r2", "c2"]
```

Out [2]:
```
20
```

In [3]:
```
df.loc["r2", :]
```

Out [3]:
```
c1      2
c2     20
c3    200
Name: r2, dtype: int64
```

In [4]:
```
df.loc[:, "c2"]
```

Out [4]:
```
r1    10
r2    20
r3    30
Name: c2, dtype: int64
```

데이터 선택하기

❖ 슬라이스나 리스트를 넘겨주는 방법

```
In [5]: df.loc[["r1", "r3"], "c2":"c3"]
```

Out [5]:

	c2	c3
r1	10	100
r3	30	300

❖ iloc 사용 방법

```
In [6]: df.iloc[1:3, [0, 2]]
```

Out [6]:

	c1	c3
r2	2	200
r3	3	300

❖ 열 이름 이용

```
In [7]: df["c2"]
```

Out [7]:

```
r1    10
r2    20
r3    30
Name: c2, dtype: int64
```

데이터 선택하기

❖ 논리값 사용 방법

```
In [8]: df > 10
```

Out [8]:

	c1	c2	c3
r1	False	False	True
r2	False	True	True
r3	False	True	True

```
In [9]: df.loc[df["c2"] > 10]
```

Out [9]:

	c1	c2	c3
r2	2	20	200
r3	3	30	300

```
In [10]: df.loc[(df["c1"] > 1) & (df["c3"] < 300)]
```

Out [10]:

	c1	c2	c3
r2	2	20	200

3

Data Cleanup:
Investigation, Matching, and Formatting

1. Query and replace
2. Finding Outliers and Bad Data
3. Formatting Data

학습목표

- Gathering Data 후 작업 후 DataFrame에 적재한 데이터 정제에 대해 알 수 있다.
- Investigation를 할 수 있다.
- Matching 를 할 수 있다.
- Formatting를 할 수 있다.

Unit 1

Query and replace

개요

- ❖ Why Clean Data?
- ❖ 일부 데이터는 올바르게 형식화되어 있어 바로 사용할 수 있지만, 대부분의 데이터는 형식 불일치 혹은 가독성 문제(예: 약어 또는 일치하지 않는 헤더 설명)가 있다.
- ❖ 둘 이상의 데이터 세트에서 데이터를 사용하는 경우 특히 심하다.
- ❖ 따라서 데이터를 정리하면 더 쉽게 저장, 검색 및 재사용을 할 수 있다.
- ❖ 데이터를 가져오고 난 후, 데이터를 새로운 데이터 형식으로 수정하고 표준화하는 것이 데이터 정리(Data Cleanup)

In [10]:
```
# 라이브러리를 임포트합니다.
import pandas as pd

# 데이터를 적재합니다.
dataframe = pd.read_csv("bigStudy/titanic.csv")

# 두 개의 행을 확인합니다.
dataframe.head(2)
```

Out [10]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

❖ 첫 번째 행과 세 개의 행, 네 개의 행을 선택

In [19]:
```
# 첫 번째 행을 선택합니다.
dataframe.iloc[0]
```

Out [19]:
```
PassengerId                     1
Survived                        0
Pclass                          3
Name            Braund, Mr. Owen Harris
Gender                         male
Age                              22
SibSp                             1
Parch                             0
Ticket                    A/5 21171
Fare                           7.25
Cabin                           NaN
Embarked                          S
Name: 0, dtype: object
```

In [20]:
```
# 세 개의 행을 선택합니다.
dataframe.iloc[1:4]
```

Out [20]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S

In [21]:
```
# 네 개의 행을 선택합니다.
dataframe.iloc[:4]
```

Out [21]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S

Query

❖ 인덱스를 이용해 조회하기

In [23]:
```
# 인덱스를 설정합니다.
dataframe = dataframe.set_index(dataframe['Name'])

# 값을 확인합니다.
dataframe.loc['Braund, Mr. Owen Harris']
```

Out [23]:
```
PassengerId                      1
Survived                         0
Pclass                           3
Name           Braund, Mr. Owen Harris
Gender                          male
Age                               22
SibSp                              1
Parch                              0
Ticket                        A/5 21171
Fare                           7.25
Cabin                           NaN
Embarked                          S
Name: Braund, Mr. Owen Harris, dtype: object
```

In [28]:
```
# dataframe[:2]와 동일합니다.
dataframe[:'Braund, Mr. Owen Harris']
```

Out [28]:

Name	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Braund, Mr. Owen Harris	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S

In [34]:
```
dataframe[['Age', 'Gender']].head(2)
```

Out [34]:

Name	Age	Gender
Braund, Mr. Owen Harris	22.0	male
Cumings, Mrs. John Bradley (Florence Briggs Thayer)	38.0	female

Query

❖ 조건에 따라 행 조회

In [35]:
```
# 'Gender' 열이 'female'인 행 중 처음 두 개를 출력합니다.
dataframe[dataframe['Gender'] == 'female'].head(2)
```

Out [35]:

Name	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Cumings, Mrs. John Bradley (Florence Briggs Thayer)	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
Heikkinen, Miss. Laina	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

In [38]:
```
# 행을 필터링합니다.
dataframe[(dataframe['Gender'] == 'female') & (dataframe['Age'] >= 60)]
```

Out [38]:

Name	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Andrews, Miss. Kornelia Theodosia	276	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0	1	0	13502	77.9583	D7	S
Warren, Mrs. Frank Manley (Anna Sophia Atkinson)	367	1	1	Warren, Mrs. Frank Manley (Anna Sophia Atkinson)	female	60.0	1	0	110813	75.2500	D37	C
Turkula, Mrs. (Hedwig)	484	1	3	Turkula, Mrs. (Hedwig)	female	63.0	0	0	4134	9.5875	NaN	S
Stone, Mrs. George Nelson (Martha Evelyn)	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0000	B28	NaN

Query and replace

Replacing

- ❖ 데이터 프레임에 있는 값을 바꾸어야 하는 경우
- ❖ replace 메서드 사용

In [43]:
```
# 값을 치환하고 두 개의 행을 출력합니다.
dataframe['Gender'].replace("female", "Woman").head(2)
```

Out [43]:
```
0     male
1    Woman
Name: Gender, dtype: object
```

In [44]:
```
# 'female'과 'male'을 'Woman'과 'Man'으로 치환합니다.
dataframe['Gender'].replace(["female", "male"], ["Woman", "Man"]).head(5)
```

Out [44]:
```
0      Man
1    Woman
2    Woman
3    Woman
4      Man
Name: Gender, dtype: object
```

In [45]:
```
# 값을 치환하고 두 개의 행을 출력합니다.
dataframe.replace(1, "One").head(2)
```

Out [45]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	One	0	3	Braund, Mr. Owen Harris	male	22	One	0	A/5 21171	7.2500	NaN	S
1	2	One	One	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	One	0	PC 17599	71.2833	C85	C

Query and replace

Replacing headers

❖ rename 메서드 사용

In [49]:
```
# 열 이름을 바꾸고 두 개의 행을 출력합니다.
dataframe.rename(columns={'PClass': 'Passenger Class'}).head(2)
```

Out [49]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

In [50]:
```
# 라이브러리를 임포트합니다.
import collections

# 딕셔너리를 만듭니다.
column_names = collections.defaultdict(str)

# 키를 만듭니다.
for name in dataframe.columns:
    column_names[name]

# 딕셔너리를 출력합니다.
column_names
```

Out [50]:
```
defaultdict(str,
            {'PassengerId': '',
             'Survived': '',
             'Pclass': '',
             'Name': '',
             'Gender': '',
             'Age': '',
             'SibSp': '',
             'Parch': '',
             'Ticket': '',
             'Fare': '',
             'Cabin': '',
             'Embarked': ''})
```

Query and replace

Replacing headers

❖ rename 메서드 사용

In [51]:
```
# 인덱스 0을 -1로 바꿉니다.
dataframe.rename(index={0:-1}).head(2)
```

Out [51]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
-1	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

In [52]:
```
# 열 이름을 소문자로 바꿉니다.
dataframe.rename(str.lower, axis='columns').head(2)
```

Out [52]:

	passengerid	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	cabin	embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

Unit 2

Finding Outliers and Bad Data

고유한 값 조회

- ❖ 열에서 고유한 값 조회
- ❖ unique 메서드 사용해 열에서 고유한 값 조회
- ❖ value_counts 메서드 사용한 고유한 값과 등장 횟수 조회

In [22]:
```
# 라이브러리를 임포트합니다.
import pandas as pd

# 데이터를 적재합니다.
dataframe = pd.read_csv("bigStudy/titanic.csv")

# 두 개의 행을 확인합니다.
dataframe.head(2)
```

Out [22]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

In [24]:
```
# 고유한 값을 찾습니다.
dataframe['Gender'].unique()
```

Out [24]:
```
array(['male', 'female'], dtype=object)
```

In [25]:
```
# 카운트를 출력합니다.
dataframe['Gender'].value_counts()
```

Out [25]:
```
male      577
female    314
Name: Gender, dtype: int64
```

고유한 값 조회

```
In [27]: # 카운트를 출력합니다.
         dataframe['Pclass'].value_counts()

Out[27]: 3    491
         1    216
         2    184
         Name: Pclass, dtype: int64

In [28]: # 고유한 값의 개수를 출력합니다.
         dataframe['Pclass'].nunique()

Out[28]: 3

In [29]: dataframe.nunique()

Out[29]: PassengerId    891
         Survived         2
         Pclass           3
         Name           891
         Gender           2
         Age             88
         SibSp            7
         Parch            7
         Ticket         681
         Fare           248
         Cabin          147
         Embarked         3
         dtype: int64
```

누락된 값 값 조회

- ❖ 데이터프레임에서 누락된 값 조회
- ❖ isnull 과 notnull

In [30]:
```
## 누락된 값을 선택하고 두 개의 행을 출력합니다.
dataframe[dataframe['Age'].isnull()].head(2)
```

Out [30]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000	NaN	S

누락된 값 값 조회

❖ NaN을 사용하려면 넘파이 라이브러리를 임포트해야 함

In [32]:
```
# NaN으로 값을 바꾸려고 합니다.
dataframe['Gender'] = dataframe['Gender'].replace('male', NaN)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-32-42cc917624a5> in <module>
      1 # NaN으로 값을 바꾸려고 합니다.
----> 2 dataframe['Gender'] = dataframe['Gender'].replace('male', NaN)

NameError: name 'NaN' is not defined
```

In [34]:
```
# 라이브러리를 임포트합니다.
import numpy as np

# NaN으로 값을 바꿉니다.
dataframe['Gender'] = dataframe['Gender'].replace('male', np.nan)
```

In [35]:
```
# 데이터를 적재하고 누락된 값을 설정합니다.
dataframe = pd.read_csv("bigStudy/titanic.csv", na_values=[np.nan, 'NONE', -999])
```

누락된 값 값 조회

❖ NaN을 사용하려면 넘파이 라이브러리를 임포트해야 함

In [36]:
```
dataframe = pd.read_csv("bigStudy/titanic.csv", na_values=['female'],
                        keep_default_na=False)
dataframe[12:14]
```

Out [36]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
12	13	0	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2151	8.050		S
13	14	0	3	Andersson, Mr. Anders Johan	male	39	1	5	347082	31.275		S

In [37]:
```
dataframe = pd.read_csv("bigStudy/titanic.csv", na_filter=False)
dataframe[12:14]
```

Out [37]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
12	13	0	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2151	8.050		S
13	14	0	3	Andersson, Mr. Anders Johan	male	39	1	5	347082	31.275		S

열 삭제

❖ 데이터프레임에서 열을 삭제하기 위해서는 drop 메서드에 axis=1 사용

In [38]:
```
# 열을 삭제합니다.
dataframe.drop('Age', axis=1).head(2)
```

Out [38]:

	PassengerId	Survived	Pclass	Name	Gender	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	1	0	A/5 21171	7.2500		S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	1	0	PC 17599	71.2833	C85	C

In [39]:
```
# 열을 삭제합니다.
dataframe.drop(['Age', 'Gender'], axis=1).head(2)
```

Out [39]:

	PassengerId	Survived	Pclass	Name	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	1	0	A/5 21171	7.2500		S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	0	PC 17599	71.2833	C85	C

In [40]:
```
# Pclass 열을 삭제합니다.
dataframe.drop(dataframe.columns[1], axis=1).head(2)
```

Out [40]:

	PassengerId	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500		S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	C85	C

In [41]:
```
dataframe.head(2)
```

Out [41]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500		S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	C85	C

행 삭제

❖ 데이터프레임에서 행을 삭제하기 위해서는 불리언 조건을 사용하여 삭제하고 싶은 행을 제외한 새로운 데이터 프레임을 생성하면 됨

In [43]:
```
# 라이브러리를 임포트합니다.
import pandas as pd

# 데이터를 적재합니다.
dataframe = pd.read_csv("bigStudy/titanic.csv")

# 두 개의 행을 확인합니다.
dataframe.head(2)
```

Out [43]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

행 삭제

❖ 데이터프레임에서 행을 삭제하기 위해서는 불리언 조건을 사용하여 삭제하고 싶은 행을 제외한 새로운 데이터 프레임을 생성하면 됨

In [45]:
```
# 행을 삭제하고 처음 두 개의 행을 출력합니다.
dataframe[dataframe['Gender'] != 'male'].head(2)
```

Out [45]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

In [46]:
```
# 행을 삭제하고 처음 두 개의 행을 출력합니다.
dataframe[dataframe['Name'] != 'Heikkinen, Miss, Laina']].head(2)
```

Out [46]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

In [47]:
```
# 행을 삭제하고 처음 두 개의 행을 출력합니다.
dataframe[dataframe.index != 0].head(2)
```

Out [47]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

중복된 행 삭제

❖ 데이터프레임에서 중복된 행을 삭제하기 위해서는 drop_duplicates을 사용

In [50]:
```
# 중복 행을 삭제하고 처음 두 개의 행을 출력합니다.
dataframe.drop_duplicates().head(2)
```

Out [50]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

In [51]:
```
# 행의 개수를 출력합니다.
print("원본 데이터프레임 행의 수:", len(dataframe))
print("중복 삭제 후 행의 수:", len(dataframe.drop_duplicates()))
```

```
원본 데이터프레임 행의 수: 891
중복 삭제 후 행의 수: 891
```

In [52]:
```
# 중복된 행을 삭제합니다.
dataframe.drop_duplicates(subset=['Gender'])
```

Out [52]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

In [53]:
```
# 중복된 행을 삭제합니다.
dataframe.drop_duplicates(subset=['Gender'], keep='last')
```

Out [53]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

값에 따라 행을 그룹핑하기

❖ 값을 기준으로 개별 행을 그룹핑하고자 할 때, groupby 사용

In [55]:
```
# 'Gender' 열의 값으로 행을 그룹핑하고 평균을 계산합니다.
dataframe.groupby('Gender').mean()
```

Out [55]:

Gender	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
female	431.028662	0.742038	2.159236	27.915709	0.694268	0.649682	44.479818
male	454.147314	0.188908	2.389948	30.726645	0.429809	0.235702	25.523893

In [56]:
```
# 행을 그룹핑합니다.
dataframe.groupby('Gender')
```

Out [56]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f71f2c16e50>
```

In [57]:
```
# 행을 그룹핑하고 카운팅합니다.
dataframe.groupby('Survived')['Name'].count()
```

Out [57]:
```
Survived
0    549
1    342
Name: Name, dtype: int64
```

In [58]:
```
# 행을 그룹핑한 다음 평균을 계산합니다.
dataframe.groupby(['Gender','Survived'])['Age'].mean()
```

Out [58]:
```
Gender  Survived
female  0           25.046875
        1           28.847716
male    0           31.618056
        1           27.276022
Name: Age, dtype: float64
```

시간에 따라 행을 그룹핑하기

❖ 시간을 기준으로 개별 행을 그룹핑하고자 할 때, resample 사용

In [59]:
```
# 라이브러리를 임포트합니다.
import pandas as pd
import numpy as np

# 날짜 범위를 만듭니다.
time_index = pd.date_range('01/01/2021', periods=100000, freq='30S')

# 데이터프레임을 만듭니다.
dataframe = pd.DataFrame(index=time_index)

# 난수 값으로 열을 만듭니다.
dataframe['Sale_Amount'] = np.random.randint(1, 10, 100000)

# 주 단위로 행을 그룹핑한 다음 합을 계산합니다.
dataframe.resample('W').sum()
```

Out [59]:

	Sale_Amount
2021-01-03	43475
2021-01-10	100698
2021-01-17	100387
2021-01-24	100954
2021-01-31	100708
2021-02-07	53364

In [60]:
```
# 처음의 행을 출력합니다.
dataframe.head(3)
```

Out [60]:

	Sale_Amount
2021-01-01 00:00:00	7
2021-01-01 00:00:30	5
2021-01-01 00:01:00	8

열 원소를 순회하기

❖ 데이터프레임의 열을 파이썬의 시퀀스처럼 쓸 수 있음

In [65]:
```
# 라이브러리를 임포트합니다.
import pandas as pd

# 데이터를 적재합니다.
dataframe = pd.read_csv("bigStudy/titanic.csv")

# 두 개의 행을 확인합니다.
dataframe.head(2)
```

Out [65]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

In [66]:
```
# 처음 두 이름을 대문자로 바꾸어 출력합니다.
for name in dataframe['Name'][0:2]:
    print(name.upper())
```

```
BRAUND, MR. OWEN HARRIS
CUMINGS, MRS. JOHN BRADLEY (FLORENCE BRIGGS THAYER)
```

In [67]:
```
# 처음 두 이름을 대문자로 바꾸어 출력합니다.
[name.upper() for name in dataframe['Name'][0:2]]
```

Out [67]:
```
['BRAUND, MR. OWEN HARRIS',
 'CUMINGS, MRS. JOHN BRADLEY (FLORENCE BRIGGS THAYER)']
```

모든 열 원소에 함수 적용

❖ 열에 있는 모든 원소에 함수를 적용하고자 할 때에는 apply를 사용

In [77]:
```
# 라이브러리를 임포트합니다.
import pandas as pd

# 데이터를 적재합니다.
dataframe = pd.read_csv("bigStudy/titanic.csv")

# 두 개의 행을 확인합니다.
dataframe.head(2)
```

Out [77]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

In [78]:
```
# 함수를 만듭니다.
def uppercase(x):
    return x.upper()

# 함수를 적용하고 두 개의 행을 출력합니다.
dataframe['Name'].apply(uppercase)[0:2]
```

Out [78]:
```
0                          BRAUND, MR. OWEN HARRIS
1    CUMINGS, MRS. JOHN BRADLEY (FLORENCE BRIGGS TH...
Name: Name, dtype: object
```

In [79]:
```
# Survived 열의 1을 Live로, 0을 Dead로 바꿉니다.
dataframe['Survived'].map({1:'Live', 0:'Dead'})[:5]
```

Out [79]:
```
0    Dead
1    Live
2    Live
3    Live
4    Dead
Name: Survived, dtype: object
```

모든 열 원소에 함수 적용

❖ 열에 있는 모든 원소에 함수를 적용하고자 할 때에는 apply를 사용

In [97]:
```
# 함수의 매개변수(age)를 apply 메서드를 호출할 때 전달할 수 있습니다.
dataframe['Age'].apply(lambda x, age: x < age, age=30)[:5]
```

Out [97]:
```
0     True
1    False
2     True
3    False
4    False
Name: Age, dtype: bool
```

In [98]:
```
def truncate_string(x):
    if type(x) == str:
        return x[:20]
    return x

# 문자열의 길이를 최대 20자로 줄입니다.
dataframe.applymap(truncate_string)[:5]
```

Out [98]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Har	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John B	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Lai	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacqu	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William H	male	35.0	0	0	373450	8.0500	NaN	S

그룹에 함수 적용하기

❖ 그룹핑을 하고 각 그룹에 함수 적용 시, groupby와 apply를 사용

In [99]:
```
# 성을 그룹핑한 다음 함수를 적용합니다.
dataframe.groupby('Gender').apply(lambda x: x.count())
```

Out [99]:

Gender	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
female	314	314	314	314	314	261	314	314	314	314	97	312
male	577	577	577	577	577	453	577	577	577	577	107	577

행 방향으로 데이터프레임을 연결

❖ concat 함수에 axis=0 매개변수를 설정하여 행의 축을 따라 연결

In [100]:
```
# 라이브러리를 임포트합니다.
import pandas as pd

# 데이터프레임을 만듭니다.
data_a = {'id': ['1', '2', '3'],
          'first': ['Alex', 'Amy', 'Allen'],
          'last': ['Anderson', 'Ackerman', 'Ali']}
dataframe_a = pd.DataFrame(data_a, columns = ['id', 'first', 'last'])

# 데이터프레임을 만듭니다.
data_b = {'id': ['4', '5', '6'],
          'first': ['Billy', 'Brian', 'Bran'],
          'last': ['Bonder', 'Black', 'Balwner']}
dataframe_b = pd.DataFrame(data_b, columns = ['id', 'first', 'last'])

# 행 방향으로 데이터프레임을 연결합니다.
pd.concat([dataframe_a, dataframe_b], axis=0)
```

Out [100]:

	id	first	last
0	1	Alex	Anderson
1	2	Amy	Ackerman
2	3	Allen	Ali
0	4	Billy	Bonder
1	5	Brian	Black
2	6	Bran	Balwner

열 방향으로 데이터프레임을 연결

❖ concat 함수에 axis=1 매개변수를 설정하여 행의 축을 따라 연결

In [2]:
```
# 열 방향으로 데이터프레임을 연결합니다.
pd.concat([dataframe_a, dataframe_b], axis=1)
```

Out [2]:

	id	first	last	id	first	last
0	1	Alex	Anderson	4	Billy	Bonder
1	2	Amy	Ackerman	5	Brian	Black
2	3	Allen	Ali	6	Bran	Balwner

❖ append를 이용한 새로운 행 추가

In [3]:
```
# 행을 만듭니다.
row = pd.Series([10, 'Chris', 'Chillon'], index=['id', 'first', 'last'])

# 행을 추가합니다.
dataframe_a.append(row, ignore_index=True)
```

Out [3]:

	id	first	last
0	1	Alex	Anderson
1	2	Amy	Ackerman
2	3	Allen	Ali
3	10	Chris	Chillon

두 데이터프레임 병합

❖ on 매개변수에 병합 열을 지정하여 merge 메서드 사용

In [4]:
```
# 라이브러리를 임포트합니다.
import pandas as pd

# 데이터프레임을 만듭니다.
employee_data = {'employee_id': ['1', '2', '3', '4'],
                 'name': ['Amy Jones', 'Allen Keys', 'Alice Bees',
                 'Tim Horton']}
dataframe_employees = pd.DataFrame(employee_data, columns = ['employee_id',
                                                             'name'])

# 데이터프레임을 만듭니다.
sales_data = {'employee_id': ['3', '4', '5', '6'],
              'total_sales': [23456, 2512, 2345, 1455]}
dataframe_sales = pd.DataFrame(sales_data, columns = ['employee_id',
                                                      'total_sales'])

# 데이터프레임을 병합합니다.
pd.merge(dataframe_employees, dataframe_sales, on='employee_id')
```

Out [4]:

	employee_id	name	total_sales
0	3	Alice Bees	23456
1	4	Tim Horton	2512

In [5]:
```
# 데이터프레임을 병합합니다.
pd.merge(dataframe_employees, dataframe_sales, on='employee_id', how='outer')
```

Out [5]:

	employee_id	name	total_sales
0	1	Amy Jones	NaN
1	2	Allen Keys	NaN
2	3	Alice Bees	23456.0
3	4	Tim Horton	2512.0
4	5	NaN	2345.0
5	6	NaN	1455.0

두 데이터프레임 병합

❖ on 매개변수에 병합 열을 지정하여 merge 메서드 사용

In [6]:
```
# 데이터프레임을 병합합니다.
pd.merge(dataframe_employees, dataframe_sales, on='employee_id', how='left')
```

Out [6]:

	employee_id	name	total_sales
0	1	Amy Jones	NaN
1	2	Allen Keys	NaN
2	3	Alice Bees	23456.0
3	4	Tim Horton	2512.0

In [7]:
```
# 데이터프레임을 병합합니다.
pd.merge(dataframe_employees,
         dataframe_sales,
         left_on='employee_id',
         right_on='employee_id')
```

Out [7]:

	employee_id	name	total_sales
0	3	Alice Bees	23456
1	4	Tim Horton	2512

Unit 3

Formatting Data

개요

- 파이썬과 같은 프로그래밍 언어에서는 글자를 문자열(string)이라고 부른다. 파이썬에서 문자열을 만들 때는 따옴표를 사용한다. 따옴표에는 큰 따옴표와 작은 따옴표가 있으며 시작 따옴표와 종료 따옴표만 같으면 어느 것을 사용해도 상관없다.
- 문자열을 출력하려면 print 명령을 사용한다.

```
[ ]  print("Hello!")

     Hello!

[ ]  print('한글도 쓸 수 있어요.')

     한글도 쓸 수 있어요.
```

문자열 연산

❖ 문자열도 숫자처럼 덧셈과 곱셈 연산을 할 수 있다. 덧셈 연산은 두 문자열을 붙이고 곱셈 연산은 문자열을 반복한다.

```
[ ]  print("내 이름은 " + "홍길동" + "입니다.")

     내 이름은 홍길동입니다.

[ ]  print("*" * 10)

     **********
```

숫자를 문자열로 바꾸기

❖ 숫자를 문자열과 더하려면 str 명령을 써서 숫자를 문자열 자료형으로 바꾸어야 한다.

```
[ ]  n = 10
     print("별표를 " + str(n) + "번 출력합니다.")
     print("*" * n)
```

```
별표를 10번 출력합니다.
**********
```

한 줄 띄우기

❖ print 명령은 한 번 호출할 때마다 한 줄씩 출력한다. 만약 print 명령을 한 번만 쓰면서 여러 줄에 걸쳐 출력을 하고 싶으면 문자열에 "다음 줄 넘기기(line feed) 기호"인 ₩n를 넣어야 한다.

```
[ ]  print("한 줄 쓰고₩n그 다음 줄을 쓴다.")
```

```
한 줄 쓰고
그 다음 줄을 쓴다.
```

줄을 바꾸지 않고 이어서 출력하기

❖ 반대로 print 명령을 여러번 쓰면서 줄은 바꾸지 않고 싶다면 다음과 같이 print 명령에 end=""이라는 인수를 추가한다.

```
[ ]  print("한 줄 쓰고 ", end="")
     print("이어서 쓴다.")

     한 줄 쓰고 이어서 쓴다.
```

문자열 값을 가지는 변수

❖ 변수에는 숫자뿐만 아니라 문자열도 넣을 수 있다.

```
[ ]  name = "홍길동"
     print("내 이름은 " + name + "입니다.")
```

```
내 이름은 홍길동입니다.
```

```
[ ]  mark = "$"
     n = 20
     print(mark + " 기호를 " + str(n) + "번 출력합니다.")
     print(mark * n)
```

```
$ 기호를 20번 출력합니다.
$$$$$$$$$$$$$$$$$$$$
```

따옴표를 출력하기

❖ 파이썬에서 두 가지 종류의 다른 따옴표를 쓸 수 있는 이유는 문자열이 따옴표를 포함하는 경우가 있기 때문이다. 만약 따옴표로 둘러싸인 문자열에 따옴표가 포함되어 있다면 파이썬은 그 부분에서 문자열이 끝난다고 인식하여 오류가 발생한다. 이처럼 문자열 안에 큰따옴표가 있어야 할 때는 전체 문자열을 작은따옴표로 둘러싸면 된다.

```
[ ]  print('둘리가 "호이!"하고 말했어요.')

     둘리가 "호이!"하고 말했어요.
```

반대로 문자열 안에 작은 따옴표가 있어야 할 때는 전체 문자열을 큰 따옴표로 둘러싼다.

```
[ ]  print("둘리가 '이제 어디로 가지?'하고 생각했어요.")

     둘리가 '이제 어디로 가지?'하고 생각했어요.
```

여러 줄의 문자열 출력하기

❖ 파이썬에서 여러 줄의 문자열을 출력하거나 변수에 할당하려면, "문자" 나 '문자' 대신 """ 여러 줄의 문자열 """ 혹은 '''여러 줄의 문자열''' 을 사용하면 된다.

```
[ ]  multi_line_string = """
     파이썬(영어: Python)은 1991년 프로그래머인
     귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로,
     플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed)
     대화형 언어이다. 파이썬이라는 이름은 귀도가 좋아하는 코미디 〈Monty Python's Flying
     Circus〉 에서 따온 것이다."""

     print(multi_line_string)
```

```
파이썬(영어: Python)은 1991년 프로그래머인
귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로,
플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed)
대화형 언어이다. 파이썬이라는 이름은 귀도가 좋아하는 코미디 〈Monty Python's Flying
Circus〉 에서 따온 것이다.
```

문자열 치환

❖ 문자열에서 특정 문자를 다른 문자로 바꾸려면 replace 메서드를 사용한다.

```
[ ]  "2020.10.23".replace(".", "-")

     '2020-10-23'
```

❖ 문자열의 공백을 없애려면 " " 공백 문자열을 "" 빈 문자열로 바꾸면 된다.

```
[ ]  "word with space".replace(" ", "")

     'wordwithspace'
```

개요

❖ 파이썬에서는 복잡한 문자열 출력을 위한 문자열 형식화(string formatting)를 지원한다. 문자열을 형식화하는 방법에는 % 기호를 사용한 방식과 format 메서드를 사용한 방식, 그리고 f 문자열을 사용하는 방식이 있다.

% 기호를 사용한 문자열 형식화

❖ 문자열 뒤에 % 기호를 붙이고 그 뒤에 다른 값을 붙이면 뒤에 붙은 값이 문자열 안으로 들어간다.

"문자열" % 값

❖ 이 때 문자열의 어느 위치에 값이 들어가는지를 표시하기 위해 문자열 안에 % 기호로 시작하는 형식지정 문자열(format specification string)을 붙인다. 대표적인 형식지정 문자열은 다음과 같다.

형식지정 문자열	의미
%s	문자열
%d	정수
%f	부동소수점 실수

Unit 2

% 기호를 사용한 문자열 형식화

다음은 % 기호를 사용한 문자열 형식의 예이다.

```
[ ]  "내 이름은 %s입니다." % "홍길동"
```

```
[ ]  "나는 %d살 입니다." % 12
```

```
'나는 12살 입니다.'
```

```
[ ]  "원주율의 값은 %f입니다." % 3.141592
```

```
'원주율의 값은 3.141592입니다.'
```

Unit 2

% 기호를 사용한 문자열 형식화

만약 여러개의 값을 문자열 안에 넣어야 한다면 % 기호 뒤에 있는 값을 소괄호로 감싸야 한다.

```
[ ]  "%d 곱하기 %d은 %d이다." % (2, 3, 6)
```

```
'2 곱하기 3은 6이다.'
```

```
[ ]  "%s의 %s 과목 점수는 %d점이다." % ("철수", "수학", 100)
```

```
'철수의 수학 과목 점수는 100점이다.'
```

고급 형식지정 문자열

형식지정 문자열은 여러가지 숫자 인수를 가질 수도 있다. % 기호 다음에 오는 정수는 값이 인쇄될 때 차지하는 공간의 길이를 뜻한다. 만약 공간의 길이가 인쇄될 값보다 크면 정수가 양수일 때는 값을 뒤로 보내고 공백을 앞에 채우거나 반대로 정수가 음수이면 값을 앞으로 보내고 공백을 뒤에 채운다. 만약 % 기호 다음에 소숫점이 있는 숫자가 오면 점 뒤의 숫자는 실수를 인쇄할 때 소숫점 아래로 그만큼의 숫자만 인쇄하라는 뜻이다.

고급 형식지정 문자열	의미
%20s	전체 20칸을 차지하는 문자열(공백을 앞에 붙인다.)
%-10d	전체 10칸을 차지하는 숫자(공백을 뒤에 붙인다.)
%.5f	부동소수점의 소수점 아래 5자리까지 표시

고급 형식지정 문자열

```
[ ]  "[%20s]" % "*"   # [와 ] 사이에 20칸의 공백이 있다.
```

```
'[                   *]'
```

```
[ ]  "[%-20s]" % "A"  # 20칸의 공백의 앞쪽에 A를 인쇄한다.
```

```
'[A                   ]'
```

```
[ ]  "[%20d]" % 123  # 20칸의 공백의 뒷쪽에 123을 인쇄한다.
```

```
'[                 123]'
```

```
[ ]  "[%-20d]" % 123  # 20칸의 공백의 앞쪽에 123을 인쇄한다.
```

```
'[123                 ]'
```

```
[ ]  x = 1 / 3.0  # 값은 0.3333333....
     "%.5f" % x   # 소숫점 아래 5자리까지만 인쇄한다.
```

```
'0.33333'
```

고급 형식지정 문자열

❖ 공백 크기와 소숫점 아래 자릿수를 같이 지정할 수도 있다.

```
[ ]  "[%-20.6f]" % x  # 20칸의 공백의 앞쪽에 소숫점 아래 6자리 출력

     '[0.333333            ]'
```

format 메서드를 사용한 문자열 형식화

`format`

❖ format 메서드를 사용하여 문자열을 형식화하는 방법도 있다. 이 때는 % 기호로 시작하는 형식지정 문자열 대신 {} 기호를 사용한다. 또한 자료형을 표시할 필요가 없다. 문자열 내에서 { 문자를 출력하고 싶을 때는 {{라는 글자를 사용한다.

```
[ ]  "내 이름은 {}입니다.".format("홍길동")
```

```
'내 이름은 홍길동입니다.'
```

```
[ ]  "내 이름은 {{{}}}입니다.".format("홍길동")
```

```
'내 이름은 {홍길동}입니다.'
```

format 메서드를 사용한 문자열 형식화

❖ {} 안에 값의 순서를 지정하는 숫자를 넣을 수도 있다. 가장 앞에 있는 값은 {0}, 그 뒤의 값은 {1}, 이런 식으로 지정한다. 이 방법을 사용하면 값의 순서를 바꾸거나 같은 값을 여러번 인쇄할 수도 있다.

```
[ ]  "{2}의 {0} 점수는 {1}점입니다. {1}점! {1}점!".format("수학", 100, "철수")

     '철수의 수학 점수는 100점입니다. 100점! 100점!'
```

❖ 순서를 나타내는 숫자 대신 인수 이름을 지정할 수도 있다.

```
[ ]  "{a}점수: {x}점, {b}점수: {y}점".format(a="영어", b="수학", x=100, y=90)

     '영어점수: 100점, 수학점수: 90점'
```

❖ 만약 {}를 여러개 사용하면서 순서 숫자나 인수 이름을 지정하지 않으면 순서대로 입력된다.

```
[ ]  "{}점수: {}점, {}점수: {}점".format("영어", 100, "수학", 90)

     '영어점수: 100점, 수학점수: 90점'
```

format 메서드를 사용한 문자열 형식화

❖ format 방식에서 공백의 크기를 지정하거나 부동소수점의 소수점 아래 숫자를 지정할 때는 {}안에 : 기호를 넣고 그 뒤에 고급 형식지정 문자열을 넣는다. : 뒤에 오는 숫자는 공백의 크기를 뜻한다. <는 값을 앞쪽으로 붙이고 공백을 뒤로 붙인다. 반대로 >는 값을 앞쪽으로 붙이고 공백을 뒤로 붙인다. 소숫점의 자릿수를 지정할 때는 .(점)과 숫자, 그리고 f 글자를 사용한다. , 기호를 넣으면 영미권에서 숫자를 쓸 때 천(1000)단위마다 붙이는 쉼표(thousand comma)를 붙인다.

고급 형식지정 문자열	의미
{:>10}	전체 10칸을 차지하며 공백을 앞에 붙임 (문자열을 오른쪽에 붙여서 출력)
{:<10}	전체 10칸을 차지하며 공백을 뒤에 붙임 (문자열을 왼쪽에 붙여서 출력)
{:^10}	전체 10칸을 차지하며 공백을 앞뒤에 붙임 (문자열을 중앙에 붙여서 출력)
{:.5f}	부동소수점의 소수점 아래 5자리까지 표시
{:,}	천단위 쉼표 표시

format 메서드를 사용한 문자열 형식화

```
[ ]  "[{:<20}]".format("*")

     '[*                   ]'

[ ]  "[{:>20}]".format("*")

     '[                   *]'

[ ]  "[{:^20}]".format("*")

     '[          *          ]'

[ ]  "[{:20.5f}]".format(1 / 3)

     '[             0.33333]'

[ ]  "[{:20,}]".format(1234567890)

     '[       1,234,567,890]'
```

format 메서드를 사용한 문자열 형식화

❖ 만약 > 기호앞에 문자열을 쓰면 해당 문자열로 공백을 채운다.

고급 형식지정 문자열	의미
{:*>10}	전체 10칸을 차지하며 "*"을 앞에 붙임 (문자열을 오른쪽에 붙여서 출력)
{:*<10}	전체 10칸을 차지하며 "*"을 뒤에 붙임 (문자열을 왼쪽에 붙여서 출력)
{:*^10}	전체 10칸을 차지하며 "*"을 앞뒤에 붙임 (문자열을 중앙에 붙여서 출력)

format 메서드를 사용한 문자열 형식화

❖ 만약 > 기호앞에 문자열을 쓰면 해당 문자열로 공백을 채운다.

```
[ ]  "[{:-<20}]".format("*")
```

```
'[*-------------------]'
```

```
[ ]  "[{:->20}]".format("*")
```

```
'[-------------------*]'
```

```
[ ]  "[{:-^20}]".format("*")
```

```
'[---------*----------]'
```

format 메서드를 사용한 문자열 형식화

❖ format 방법의 또다른 특징은 {}안에서 리스트나 사전의 인덱싱을 할수 있다는 것이다.

```
[ ]  x = [10, 11, 12]
     "리스트의 첫번째 원소={0[0]}".format(x)
```

```
'리스트의 첫번째 원소=10'
```

```
[ ]  y = {"a": 10, "b": 11, "c": 12}
     "사전의 a키 값={0[a]}".format(y)
```

```
'사전의 a키 값=10'
```

f 문자열

- f 문자열(f-string)
- 파이썬 3.6부터는 f 문자열(f-string)이라는 것을 사용할 수 있다. f 문자열은 문자열의 앞에 f 글자를 붙인 문자열이다. f 문자열에서는 {} 안에 변수의 이름을 바로 사용할 수 있다.

```
[ ]  name = "홍길동"
     age = 32
     print(f"{name}의 나이는 {age}살이다.")
```

```
홍길동의 나이는 32살이다.
```

f 문자열

❖ f 문자열에서 공백의 크기 등을 지정할 때는 format 방법과 같은 고급 형식 지정 문자열을 사용할 수 있다.

```
[ ]  number = 1234567

[ ]  f"[{number:<20}]"

     '[1234567             ]'

[ ]  f"[{number:>20}]"

     '[             1234567]'

[ ]  f"[{number:^20}]"

     '[      1234567       ]'

[ ]  f"[{number:-<20}]"

     '[1234567-------------]'
```

f 문자열

❖ f 문자열에서 공백의 크기 등을 지정할 때는 format 방법과 같은 고급 형식 지정 문자열을 사용할 수 있다.

```
[ ]  f"[{number:->20}]"

     '[----------1234567]'

[ ]  f"[{number:-^20}]"

     '[------1234567-------]'

[ ]  f"[{number:-^20,}]"

     '[-----1,234,567------]'

[ ]  f_number = 3.141592

[ ]  f"[{f_number:20.3f}]"

     '[               3.142]'
```

Unit 4

실습과제

2일차-과제

샘플 데이터 셋을 만들고, 구축 과정 스크립트 작성

❖ 샘플로 제공되어지는 다음 데이터 셋은 아이폰, 안드로이드 스마트폰을 소유한 전국의 15-49세 남녀 대상 설문조사 데이터와 이것을 분석용 데이터 셋으로 구축하는 과정을 보여준다. 샘플 데이터 셋을 만들고, 구축 과정 스크립트를 작성해 제출하시오.

<채점기준표>

채점 항목	배점	결과	채점 세부항목	
2-1. 분석용 데이터 셋을 구축하는 능력	80		분석용 데이터 적재부터 구축까지 흐름을 잘 파악하고 있다.	80
			분석용 데이터 적재부터 구축까지 흐름을 잘 파악하고 있지는 않지만, 의미가 동일하게 기재하고 있다.	50
			분석용 데이터 적재부터 구축까지 흐름을 일부만 파악하고 있다.	20
			적지 않았을 경우	0
2-2. 분석용 데이터 셋을 식별하는 능력	20		분석용 데이터 셋을 잘 식별하고 있다.	20
			분석용 데이터 셋을 잘 식별하고 있지는 않지만, 의미가 동일하게 기재하고 있다.	15
			분석용 데이터 셋을 일부만 식별하고 있다.	10
			적지 않았을 경우	0
합계	100			

수행 예

샘플 데이터 셋

	스마트폰 관련			사용자 관련					
	출시년도	구매시기	디스플레이 크기 (인치)	연령	키 (cm)	몸무게 (kg)	평균 스마트폰 사용시간 (분/일)	평균 컴퓨터 사용시간 (분/일)	데이터 사용량 (mb/일)
1	2015	2015	5	45	173	75	60	500	100
2	2014	2015	4.5	27	176	59	70	30	50
3	2015	2015	5	29	183	65	120	300	200
4	2015	2016	5	28	172	63	80	60	190
5	2015	2015	5	24	179	65	90	30	500
6	2016	2016	5.5	34	175	73	80	480	160
7	2016	2016	6	40	169	77	60	300	170
8	2014	2014	5	40	182	82	50	60	50
9	2014	2014	4.5	30	175	70	100	90	300
10	2015	2015	5	42	177	79	40	480	30
11	2014	2014	4.5	32	167	62	130	300	400
12	2015	2015	5.5	37	171	70	90	360	200
13	2015	2015	5.5	35	173	69	60	120	150
14	2016	2016	6	41	181	88	60	120	130
15	2015	2015	5.5	40	178	89	70	60	140
16	2015	2015	5	35	169	70	50	90	80
17	2016	2016	5.5	35	184	72	90	90	70
18	2016	2016	6	34	176	71	90	60	130
19	2015	2015	4.5	29	188	83	140	240	600
20	2016	2016	5.5	30	175	70	50	120	50
21	2015	2015	5.5	32	177	68	70	120	300
22	2016	2016	5.5	25	182	73	60	60	300
23	2015	2015	5	26	179	80	180	30	1100
24	2016	2016	6	25	177	76	120	90	700

1.데이터 셋 로드

```
import pandas as pd
```

```
스마트폰=pd.read_excel("phone.xlsx")
```

EDA

```
[ ] 스마트폰.head()
```

	Unnamed: 0	스마트폰 관련	Unnamed: 2	Unnamed: 3	사용자 관련	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	NaN	출시년도	구매시기	디스플레이 크기₩n(인치)	연령	키₩n(cm)	몸무게₩n(kg)	평균 스마트폰₩n사용시간₩n(분/일)	평균 컴퓨터₩n사용시간₩n(분/일)	데이터₩n사용량₩n(mb/일)
1	1.0	2015	2015	5	45	173	75	60	500	100
2	2.0	2014	2015	4.5	27	176	59	70	30	50
3	3.0	2015	2015	5	29	183	65	120	300	200
4	4.0	2015	2016	5	28	172	63	80	60	190

2.EDA

```
[ ] # 행 개수
    len(스마트폰)
```

```
25
```

```
[ ] # 열 개수
    len(스마트폰.columns)
```

```
10
```

```
[ ] 스마트폰.shape
```

```
(25, 10)
```

```
[ ] # Null 값이 아닌 행 개수
    스마트폰.count()
```

```
Unnamed: 0    24
스마트폰 관련       25
Unnamed: 2    25
Unnamed: 3    25
사용자 관련        25
Unnamed: 5    25
Unnamed: 6    25
Unnamed: 7    25
Unnamed: 8    25
Unnamed: 9    25
dtype: int64
```

```
[ ] 스마트폰['Unnamed: 0'].count()
```

```
24
```

```
[ ] # 그룹별 개수 구하기
    스마트폰.groupby('Unnamed: 0').size()
    스마트폰.groupby('Unnamed: 0').count()
```

3. 필요없는 행 삭제 후 데이터 셋 저장

```
[ ] # 행 삭제
    스마트폰=스마트폰.drop(index=[0], axis=0)
    스마트폰.head()
    # 열 삭제 : 스마트폰=스마트폰.drop(columns=['col1','col2'], axis=1)
    # 조건에 맞는 행 삭제 : 스마트폰=스마트폰[스마트폰['col1'] != 15]
```

	Unnamed: 0	스마트폰 관련	Unnamed: 2	Unnamed: 3	사용자 관련	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
1	1.0	2015	2015	5	45	173	75	60	500	100
2	2.0	2014	2015	4.5	27	176	59	70	30	50
3	3.0	2015	2015	5	29	183	65	120	300	200
4	4.0	2015	2016	5	28	172	63	80	60	190
5	5.0	2015	2015	5	24	179	65	90	30	500

```
[ ] 스마트폰.shape
```

```
(24, 10)
```

```
[ ] 스마트폰.to_csv('스마트폰.csv')
```

4. 데이터 셋 로드-csv

```
import pandas as pd
```

```
[ ] 스마트폰=pd.read_csv('스마트폰.csv')
```

```
[ ] 스마트폰.shape
```

```
(23, 10)
```

```
[ ] 스마트폰.head()
```

	1	2015	2015.1	5	45	173	75	60	500	100
0	2	2014	2015	4.5	27	176	59	70	30	50
1	3	2015	2015	5.0	29	183	65	120	300	200
2	4	2015	2016	5.0	28	172	63	80	60	190
3	5	2015	2015	5.0	24	179	65	90	30	500
4	6	2016	2016	5.5	34	175	73	80	480	160

5. 열 이름 변경 및 분석 데이터 셋 저장

```
# 행,열 하나 하나에 대해서 이름을 변경
스마트폰=스마트폰.rename(
    {
        '1':'순번',
        '2015':'출시년도'
    }, axis='columns'
)
```

```
스마트폰.head()
```

	순번	출시년도	2015.1	5	45	173	75	60	500	100
0	2	2014	2015	4.5	27	176	59	70	30	50
1	3	2015	2015	5.0	29	183	65	120	300	200
2	4	2015	2016	5.0	28	172	63	80	60	190
3	5	2015	2015	5.0	24	179	65	90	30	500
4	6	2016	2016	5.5	34	175	73	80	480	160

```
# 모든 열에 대해서 한꺼번에 변경
스마트폰.columns=['순번','출시년도','구매시기', "디스플레이","연령","키","몸무게","스마트폰사용시간","컴퓨터사용시간","데이터사용량"]
```

```
스마트폰.head()
```

	순번	출시년도	구매시기	디스플레이	연령	키	몸무게	스마트폰사용시간	컴퓨터사용시간	데이터사용량
0	2	2014	2015	4.5	27	176	59	70	30	50
1	3	2015	2015	5.0	29	183	65	120	300	200
2	4	2015	2016	5.0	28	172	63	80	60	190
3	5	2015	2015	5.0	24	179	65	90	30	500
4	6	2016	2016	5.5	34	175	73	80	480	160

```
스마트폰.to_csv('스마트폰.csv')
```

6. 분석 데이터 셋 다시 적재

```
[ ] import pandas as pd
    스마트폰=pd.read_csv('스마트폰.csv')
```

```
[ ] 스마트폰.head()
```

	Unnamed: 0	순번	출시년도	구매시기	디스플레이	연령	키	몸무게	스마트폰사용시간	컴퓨터사용시간	데이터사용량
0	0	2	2014	2015	4.5	27	176	59	70	30	50
1	1	3	2015	2015	5.0	29	183	65	120	300	200
2	2	4	2015	2016	5.0	28	172	63	80	60	190
3	3	5	2015	2015	5.0	24	179	65	90	30	500
4	4	6	2016	2016	5.5	34	175	73	80	480	160

```
[ ] 스마트폰=스마트폰.drop(columns=['Unnamed: 0'], axis=1)
```

```
[ ] 스마트폰.head()
```

	순번	출시년도	구매시기	디스플레이	연령	키	몸무게	스마트폰사용시간	컴퓨터사용시간	데이터사용량
0	2	2014	2015	4.5	27	176	59	70	30	50
1	3	2015	2015	5.0	29	183	65	120	300	200
2	4	2015	2016	5.0	28	172	63	80	60	190
3	5	2015	2015	5.0	24	179	65	90	30	500
4	6	2016	2016	5.5	34	175	73	80	480	160

7. 분석 데이터 식별

1. 출시년도가 2016년도별 조회

```
[ ] 스마트폰출시년도조회=스마트폰.loc[
                        (
                            스마트폰['출시년도'] == 2016
                        )
    ]
```

```
[ ] len(스마트폰출시년도조회)
```

```
8
```

```
[ ] 스마트폰출시년도조회.head()
```

	순번	출시년도	구매시기	디스플레이	연령	키	몸무게	스마트폰사용시간	컴퓨터사용시간	데이터사용량
4	6	2016	2016	5.5	34	175	73	80	480	160
5	7	2016	2016	6.0	40	169	77	60	300	170
12	14	2016	2016	6.0	41	181	88	60	120	130
15	17	2016	2016	5.5	35	184	72	90	90	70
16	18	2016	2016	6.0	34	176	71	90	60	130

7. 분석 데이터 식별

2. 데이터 사용량이 500이상인 내용 조회

```
스마트폰데이터사용량=스마트폰.loc[
                    (
                        스마트폰['데이터사용량'] > 500
                    )
]
```

```
[ ] len(스마트폰데이터사용량)
```

```
3
```

```
[ ] 스마트폰데이터사용량.head()
```

	순번	출시년도	구매시기	디스플레이	연령	키	몸무게	스마트폰사용시간	컴퓨터사용시간	데이터사용량
17	19	2015	2015	4.5	29	188	83	140	240	600
21	23	2015	2015	5.0	26	179	80	180	30	1100
22	24	2016	2016	6.0	25	177	76	120	90	700

제출방법

형식

❖ 형식 : 보고서와 산출물로 작성된 *.csv 파일 제출(파일명 : 과제2일차-사번.txt 혹은 *.hwp 혹은 *.doc" 파일로 작성, 작성된 *.csv 파일을 압축하여 과제2일차.zip으로 만들어 제출할 것.)

Unit A

참고자료

참고자료

- http://www.ncs.go.kr
- NELLDALE/JOHN LEWIS지음, 조영석/김대경/박찬영/송창근 역, 단계별로 배우는컴퓨터과학, 홍릉과학출판사,2018
- 혼자 공부하는 머신러닝+딥러닝 박해선 지음 | 한빛미디어 | 2020년 12월
- 머신러닝 실무 프로젝트 ,아리가 미치아키, 나카야마 신타, 니시바야시 다카시 지음 | 심효섭 옮김 | 한빛미디어 | 2018년 06월
- 파이썬을 활용한 머신러닝 쿡북 크리스 알본 지음 | 박해선 옮김 | 한빛미디어 | 2019년 09월
- 처음 배우는 머신러닝 김의중 지음 | 위키북스 | 2016년 07월
- 파이썬으로 배우는 머신러닝의 교과서 : 이토 마코토 지음 | 박광수(아크몬드) 옮김 | 한빛미디어 | 2018년 11월
- 기타 서적 및 웹 사이트 자료 다수 참조

나눔고딕 인스톨

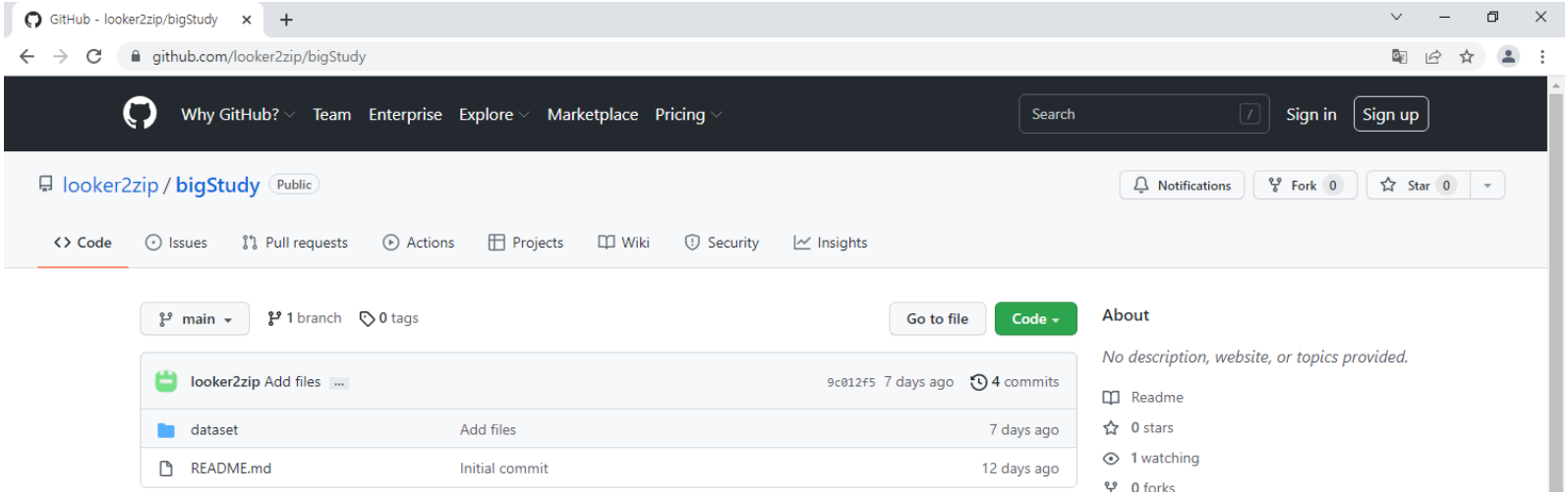
❖ 리눅스 다운로드

!rm -rf bigStudy

!git clone 'https://github.com/looker2zip/bigStudy.git'

❖ 윈도 다운로드

https://github.com/looker2zip/bigStudy



나눔고딕 인스톨

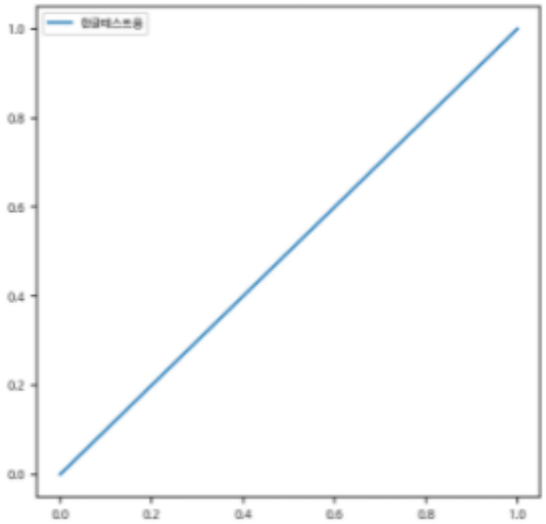
```
%config InlineBackend.figure_format = 'retina'
 !apt -qq -y install fonts-nanum
```

❖ 런타임 다시 시작

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm
fontpath = '/usr/share/fonts/truetype/nanum/NanumBarunGothic.ttf'
font = fm.FontProperties(fname=fontpath, size=9)
plt.rc('font', family='NanumBarunGothic')
mpl.font_manager._rebuild()


plt.figure(figsize=(5,5))
plt.plot([0,1], [0,1], label='한글테스트용')
plt.legend()
plt.show()
```

나눔고딕 인스톨

```
plt.figure(figsize=(5,5))
plt.plot([0,1], [0,1], label='한글테스트용')
plt.legend()
plt.show()
```



- Mobile: 010-9591-1401
- E-mail: onlooker2zip@naver.com