스마트 모빌리티 프로그래밍

Ch 11. 파이썬 확장 패키지 (2)
– Pandas, Matplotlib, Seaborn

영남대학교 정보통신공학과

교수 김 영 탁

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

Outline

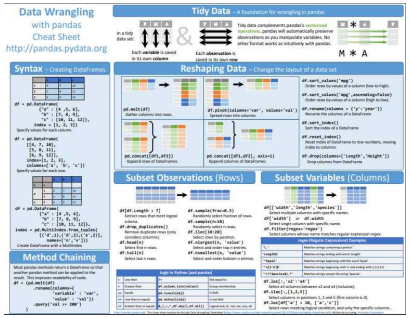
- Pandas 및 데이터 분석
- Matplotlib 기반 시각화 (Visualization)
- Seaborn 기반 시각화

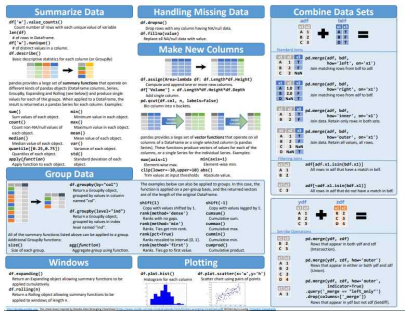
Pandas

Pandas

◆ Pandas

- 데이터 조작 및 분석에 사용되는 파이썬 라이브러리
- 재무, 경제, 통계, 광고, 웹분석 등 다양한 영역에서 사용됨
- Pandas를 사용하여 데이터 불러오기, 저장하기, 분석, 필터링, 정렬, 그룹화, 누락 데이터의 정제 등을 수행할 수 있음
- 관련자료:
  - 10 minutes to pandas - https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html
  - pandas cheat sheet - https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf





pandas 설치

◆ >python -m pip install --upgrade pandas

```
C:\Users\Owner>python -m pip install --upgrade pandas
Collecting pandas
  Downloading pandas-1.2.0-cp39-cp39-win_amd64.whl (9.3 MB)
     |████████████████████████████████| 9.3 MB 6.4 MB/s
Requirement already satisfied: numpy>=1.16.5 in c:\users\owner\appdata\local\programs\python\python39\lib\site-packages
(from pandas) (1.19.4)
Requirement already satisfied: pytz>=2017.3 in c:\users\owner\appdata\local\programs\python\python39\lib\site-packages (
from pandas) (2020.5)
Collecting python-dateutil>=2.7.3
  Using cached python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
Requirement already satisfied: six>=1.5 in c:\users\owner\appdata\local\programs\python\python39\lib\site-packages (from
 python-dateutil>=2.7.3->pandas) (1.15.0)
Installing collected packages: python-dateutil, pandas
Successfully installed pandas-1.2.0 python-dateutil-2.8.1
```

Pandas 기본 자료 구조

◆ Series

- 1차원 배열과 유사한 형태
- Python의 list나 NumPy의 array로 생성
- 값과 함께 개발자가 지정하는 인덱스 값을 설정할 수 있음
- 테이블에서 열 (column)의 데이터를 나타냄

◆ DataFrame

- 2차원 배열 형태의 테이블
- NumPy 배열이나 Python의 사전 (dict)형으로 생성
- Series의 모음으로 만들어진 테이블

Pandas Series

◆ Pandas Series 생성

```
# pandas basic - creation of series

import pandas as pd
import numpy as np

data_1 = [1, 2, 6, 7, np.nan, 9, 10, 11]
sr_1 = pd.Series(data_1)
print("sr_1 =")
print(sr_1)

data_2 = [1, 2, 6, 7]
index_2 = ['a', 'b', 'c', 'd']
sr_2 = pd.Series(data_2, index= index_2)
print("sr_2 =")
print(sr_2)

data_3 = {'a':1, 'b':2, 'c':6, 'd':7}
sr_3 = pd.Series(data_3)
print("sr_3 =")
print(sr_3)
```

```
sr_1 =
0     1.0
1     2.0
2     6.0
3     7.0
4     NaN
5     9.0
6    10.0
7    11.0
dtype: float64
sr_2 =
a    1
b    2
c    6
d    7
dtype: int64
sr_3 =
a    1
b    2
c    6
d    7
dtype: int64
```

Series 및 DataFrame 생성 예

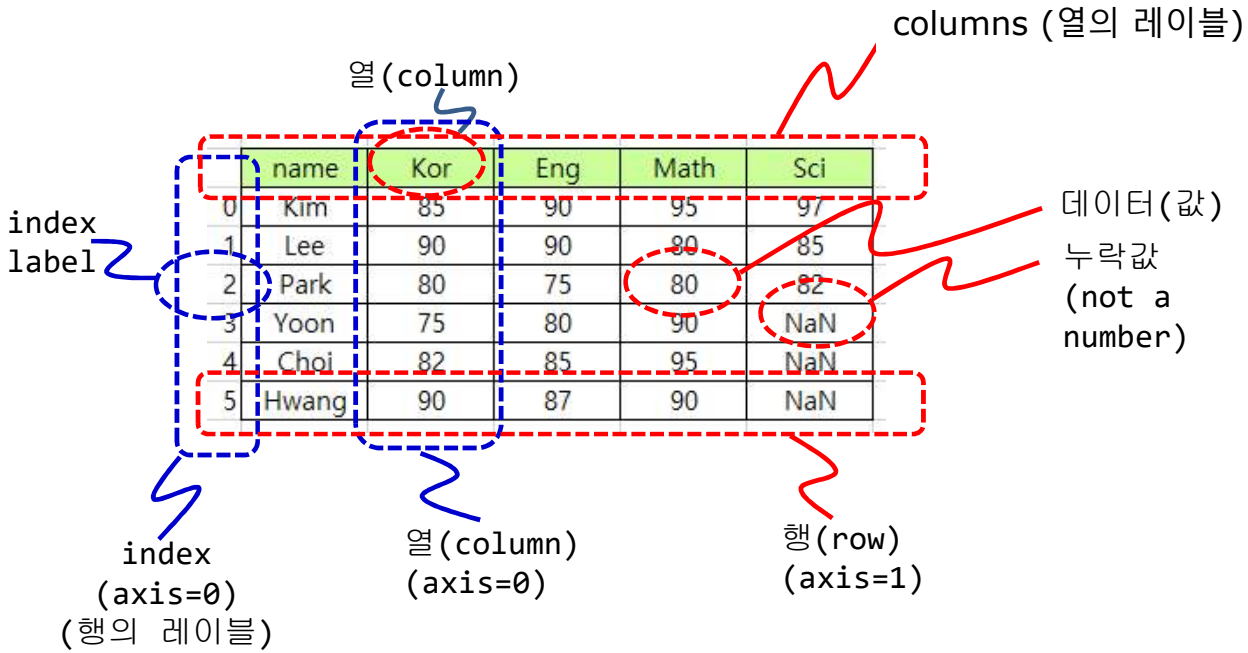
```
# pandas - creation of series

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 2300]
st_names = {"Name" : ['Kim', 'Lee', 'Park', 'Yoon', 'Choi']}

st_index = pd.Series(st_ids)
print("st_index = ")
print(st_index)

students_index_auto = pd.DataFrame(st_names)
print("students_index_auto = ")
print(students_index_auto)

students_index_stID = pd.DataFrame(st_names, index=st_ids)
print("students_index_stID = ")
print(students_index_stID)
```

```
st_index =
0    1201
1    2202
2    1203
3    1701
4    2300
dtype: int64
students_index_auto =
   Name
0   Kim
1   Lee
2  Park
3  Yoon
4  Choi
students_index_stID =
      Name
1201   Kim
2202   Lee
1203  Park
1701  Yoon
2300  Choi
```

pandas DataFrame 구조

◆ pandas DataFrame 구조

- 인덱스(index) 객체: 행 (row)의 레이블 (label)
- columns 객체: 열(column)의 레이블



	name	Kor	Eng	Math	Sci
0	Kim	85	90	95	97
1	Lee	90	90	80	85
2	Park	80	75	80	82
3	Yoon	75	80	90	NaN
4	Choi	82	85	95	NaN
5	Hwang	90	87	90	NaN

Pandas DataFrame

◆ Pandas DataFrame 생성(1)

```
# pandas basic - creation of DataFrame with date_range

import pandas as pd
import numpy as np

dates = pd.date_range("20210101", periods=5)
print(dates)

temps = [[-1, 3], [-3, 2], [-5, 5], [-2, 7], [1, 10]]
df = pd.DataFrame(temps, index=dates, columns=["low", "high"])
print("\nTemperatures = ")
print(df)
```

```
DatetimeIndex(['2021-01-01', '2021-01-02', '2021-01-03', '2021-01-04',
               '2021-01-05'],
              dtype='datetime64[ns]', freq='D')

Temperatures =
            low  high
2021-01-01   -1     3
2021-01-02   -3     2
2021-01-03   -5     5
2021-01-04   -2     7
2021-01-05    1    10
```

Pandas DataFrame

◆ Pandas DataFrame 생성(2)

```
# pandas basic - creation of DataFrame with dict

import pandas as pd
import numpy as np

df = pd.DataFrame({"Date": pd.date_range("20210101", periods=5),
                   "low": [-1, -3, -5, -2, 1],
                   "high" : [3, 2, 5, 7, 10]})
print("\nTemperatures = ")
print(df)
print(df.dtypes)
print("\ndf.info()")
print(df.info())
```

```
Temperatures =
        Date  low  high
0 2021-01-01   -1     3
1 2021-01-02   -3     2
2 2021-01-03   -5     5
3 2021-01-04   -2     7
4 2021-01-05    1    10
Date    datetime64[ns]
low              int64
high             int64
dtype: object
```

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    5 non-null      datetime64[ns]
 1   low     5 non-null      int64
 2   high    5 non-null      int64
dtypes: datetime64[ns](1), int64(2)
memory usage: 184.0 bytes
None
```

Pandas DataFrame

◆ Pandas DataFrame 보기 - head(), tail()

```
# pandas basic - creation of DataFrame with dict

import pandas as pd
import numpy as np

df = pd.DataFrame({"Date": pd.date_range("20210101", periods=10),
                   "low": [-1, -3, -5, -2, 1, 2, 1, 0, -1, -3],
                   "high" : [3, 2, 5, 7, 10, 8, 9, 7, 5, 3]})
print("\nTemperatures = ")
print(df)

print("df.head()")
print(df.head()) # default 5 first rows

print("df.tail()")
print(df.tail()) # default 5 last rows
```

```
Temperatures =
        Date  low  high
0 2021-01-01   -1     3
1 2021-01-02   -3     2
2 2021-01-03   -5     5
3 2021-01-04   -2     7
4 2021-01-05    1    10
5 2021-01-06    2     8
6 2021-01-07    1     9
7 2021-01-08    0     7
8 2021-01-09   -1     5
9 2021-01-10   -3     3
df.head()
        Date  low  high
0 2021-01-01   -1     3
1 2021-01-02   -3     2
2 2021-01-03   -5     5
3 2021-01-04   -2     7
4 2021-01-05    1    10
df.tail()
        Date  low  high
5 2021-01-06    2     8
6 2021-01-07    1     9
7 2021-01-08    0     7
8 2021-01-09   -1     5
9 2021-01-10   -3     3
```

Pandas DataFrame

◆ Pandas DataFrame의 통계적 요약정보 - describe()

```
# pandas DataFrame - describe()

import pandas as pd
import numpy as np

df = pd.DataFrame({"Date": pd.date_range("20210101", periods=10),
                   "low": [-1, -3, -5, -2, 1, 2, 1, 0, -1, -3],
                   "high" : [3, 2, 5, 7, 10, 8, 9, 7, 5, 3]})
print("\nTemperatures = ")
print(df)

print("df.describe()")
print(df.describe())
```

```
Temperatures =
        Date  low  high
0 2021-01-01   -1     3
1 2021-01-02   -3     2
2 2021-01-03   -5     5
3 2021-01-04   -2     7
4 2021-01-05    1    10
5 2021-01-06    2     8
6 2021-01-07    1     9
7 2021-01-08    0     7
8 2021-01-09   -1     5
9 2021-01-10   -3     3
df.describe()
             low       high
count  10.00000  10.000000
mean   -1.10000   5.900000
std     2.18327   2.726414
min    -5.00000   2.000000
25%    -2.75000   3.500000
50%    -1.00000   6.000000
75%     0.75000   7.750000
max     2.00000  10.000000
```

Pandas DataFrame

◆ Pandas DataFrame의 전치 행렬 (transpose)

```
#pandas DataFrame - describe()

import pandas as pd

df = pd.DataFrame({"Date": pd.date_range("20210101", periods=10),
                   "low": [-1, -3, -5, -2, 1, 2, 1, 0, -1, -3],
                   "high" : [3, 2, 5, 7, 10, 8, 9, 7, 5, 3]})
print("\nTemperatures = ")
print(df)

print("df.T")
print(df.T)
```

```
Temperatures =
        Date  low  high
0 2021-01-01   -1     3
1 2021-01-02   -3     2
2 2021-01-03   -5     5
3 2021-01-04   -2     7
4 2021-01-05    1    10
5 2021-01-06    2     8
6 2021-01-07    1     9
7 2021-01-08    0     7
8 2021-01-09   -1     5
9 2021-01-10   -3     3
df.T
                           0  ...                   9
Date  2021-01-01 00:00:00  ...  2021-01-10 00:00:00
low                    -1  ...                  -3
high                    3  ...                   3

[3 rows x 10 columns]
```

DataFrame 정렬 - sort_index(), sort_values()

```
# pandas - calculate mean and add one more column

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
        'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
        'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
        'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
        }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("after sort_index():")
print(df.sort_index(axis=0))

print("after sort_values(by='Eng', ascending=False):")
print(df.sort_values(by="Eng", ascending=False))

print("after sort_values(by='Kor', ascending=False):")
print(df.sort_values(by="Kor", ascending=False))

print("after sort_values(by='Math', ascending=False):")
print(df.sort_values(by="Math", ascending=False))
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
after sort_index():
     st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
1203    Park  85.7  88.7  90.3
1500    Choi  98.9  97.2  98.2
1701    Yoon  76.8  80.2  83.5
2202     Lee  92.4  94.5  93.5
after sort_values(by='Eng', ascending=False):
     st_name   Eng   Kor  Math
1500    Choi  98.9  97.2  98.2
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
after sort_values(by='Kor', ascending=False):
     st_name   Eng   Kor  Math
1500    Choi  98.9  97.2  98.2
2202     Lee  92.4  94.5  93.5
1201     Kim  95.7  92.3  95.2
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
after sort_values(by='Math', ascending=False):
     st_name   Eng   Kor  Math
1500    Choi  98.9  97.2  98.2
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
```

DataFrame - 열 (column) 단위 선택

```
# pandas - calculate mean and add one more column

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df['Kor']:")
print(df['Kor'])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df['Kor']:
1201    92.3
2202    94.5
1203    88.7
1701    80.2
1500    97.2
Name: Kor, dtype: float64
```

df[1:4] - 범위가 지정된 행 (row) 선택

```
# pandas - calculate mean and add one more column

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df[1:4]:")
print(df[1:4])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df[1:4]:
     st_name   Eng   Kor  Math
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
```

df.loc[index] - index를 사용한 행 (row) 선택

```
# pandas - select row with label

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("st_ids[1] = ", st_ids[1])
print("df.loc[st_ids[1]]")
print(df.loc[st_ids[1]])

print("df.loc[2202]")
print(df.loc[2202])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
st_ids[1] =  2202
df.loc[st_ids[1]]
st_name     Lee
Eng        92.4
Kor        94.5
Math       93.5
Name: 2202, dtype: object
df.loc[2202]
st_name     Lee
Eng        92.4
Kor        94.5
Math       93.5
Name: 2202, dtype: object
```

df.loc[:, [‘Eng’, ‘Math’]] - 모든 행의 지정된 열 선택

```
# pandas - select columns of all rows

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df.loc[:, ['Eng', 'Math']]")
print(df.loc[:, ['Eng', 'Math']])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df.loc[:, ['Eng', 'Math']]
       Eng  Math
1201  95.7  95.2
2202  92.4  93.5
1203  85.7  90.3
1701  76.8  83.5
1500  98.9  98.2
```

df.loc[idx_fr : idx_to, ['st_name', 'kor']] - 범위로 지정된 행의 특정 열 선택

```
# pandas - students, select specific columns of specific rows

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df.loc[2202:1701, ['st_name', 'Kor']]")
print(df.loc[2202:1701, ['st_name', 'Kor']])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df.loc[2202:1701, ['st_name', 'Kor']]
     st_name   Kor
2202     Lee  94.5
1203    Park  88.7
1701    Yoon  80.2
```

df.loc[index, label] - 특정 행, 특정 열의 값 선택

```
# pandas - students, retrieve value of given row-column

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df.loc[2202, 'Kor']")
print(df.loc[2202, 'Kor'])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df.loc[2202, 'Kor']
94.5
```

DataFrame - df.at()을 사용한 특정 행, 특정 열의 값 선택

```
# pandas - students, retrieve value of given row-column

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df.at[2202, 'Kor'] = ", df.at[2202, 'Kor'])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df.at[2202, 'Kor'] =  94.5
```

df.iloc[3] - 위치(position)를 사용한 행 선택

```
# pandas - iloc(), select row with position

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df.iloc[3]")
print(df.iloc[3])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df.iloc[3]
st_name    Yoon
Eng        76.8
Kor        80.2
Math       83.5
Name: 1701, dtype: object
```

df.iloc[2:5, 1:3] - 주어진 영역의 행, 열 선택

```
# pandas - iloc(), select with range of rows and columns

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df.iloc[2:5, 1:3]")
print(df.iloc[2:5, 1:3])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df.iloc[2:5, 1:3]
       Eng   Kor
1203  85.7  88.7
1701  76.8  80.2
1500  98.9  97.2
```

df.iloc[[1,2,4], [0, 2]] - 주어진 행 목록, 열 목록으로 선택

```
# pandas - iloc(), select with list of rows and columns

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df.iloc[[1, 2, 4], [0, 2]]")
print(df.iloc[[1, 2, 4], [0, 2]])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df.iloc[[1, 2, 4], [0, 2]]
     st_name   Kor
2202     Lee  94.5
1203    Park  88.7
1500    Choi  97.2
```

df.iloc[1:3, :] - 범위가 지정된 행 (row) 선택

```
# pandas - iloc(), select with range of rows

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df.iloc[1:3, :]")
print(df.iloc[1:3, :])
```

```
df =
      st_name   Eng   Kor  Math
1201      Kim  95.7  92.3  95.2
2202      Lee  92.4  94.5  93.5
1203     Park  85.7  88.7  90.3
1701     Yoon  76.8  80.2  83.5
1500     Choi  98.9  97.2  98.2
df.iloc[1:3, :]
     st_name   Eng   Kor  Math
2202      Lee  92.4  94.5  93.5
1203     Park  85.7  88.7  90.3
```

df.iloc[:, 2:4] - 범위가 지정된 열 (column) 선택

```
# pandas - iloc(), select range of columns

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df.iloc[:, 2:4]")
print(df.iloc[:, 2:4])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df.iloc[:, 2:4]
       Kor  Math
1201  92.3  95.2
2202  94.5  93.5
1203  88.7  90.3
1701  80.2  83.5
1500  97.2  98.2
```

df.iloc[1, 3] - 지정된 위치의 행-열의 값 추출

```
# pandas - iloc[1, 3], select with given row_column indices

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df.iloc[1, 3] = ", df.iloc[1, 3])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df.iloc[1, 3] =  93.5
```

df[df.Kor >= 90] - 주어진 조건식을 만족하는 열을 가진 행(들)만 선택

```
# pandas - df[df.Kor >= 90], select rows with conditional column value

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

print("df[df.Kor >= 90] = ")
print(df[df.Kor >= 90])
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
df[df.Kor >= 90] =
     st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1500    Choi  98.9  97.2  98.2
```

새로운 열 (column)의 추가

```
# pandas - df, addition of new column

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

sci_data = [75.9, 92.4, 87.3, 75.4, 95.3]
print("sci_data = ", sci_data)
df.loc[:, 'Sci'] = sci_data
print("df with addition of Sci = ")
print(df)
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
1500    Choi  98.9  97.2  98.2
sci_data =  [75.9, 92.4, 87.3, 75.4, 95.3]
df with addition of Sci =
     st_name   Eng   Kor  Math   Sci
1201     Kim  95.7  92.3  95.2  75.9
2202     Lee  92.4  94.5  93.5  92.4
1203    Park  85.7  88.7  90.3  87.3
1701    Yoon  76.8  80.2  83.5  75.4
1500    Choi  98.9  97.2  98.2  95.3
```

행 (학생)별 평균 계산, Avg 열 (column) 추가

```
# pandas - df, calculation of average of each class

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2],
       'Sci': [75.9, 92.4, 87.3, 75.4, 95.3]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

avgs_per_student = df.mean(1)
print("\navgs_per_student =")
print(avgs_per_student)

df.loc[:, 'Avg'] = avgs_per_student
print("\ndf_with_avg =")
print(df)
```

```
df =
      st_name   Eng   Kor  Math   Sci
1201     Kim  95.7  92.3  95.2  75.9
2202     Lee  92.4  94.5  93.5  92.4
1203    Park  85.7  88.7  90.3  87.3
1701    Yoon  76.8  80.2  83.5  75.4
1500    Choi  98.9  97.2  98.2  95.3

avgs_per_student =
1201    89.775
2202    93.200
1203    88.000
1701    78.975
1500    97.400
dtype: float64

df_with_avg =
     st_name   Eng   Kor  Math   Sci      Avg
1201     Kim  95.7  92.3  95.2  75.9  89.775
2202     Lee  92.4  94.5  93.5  92.4  93.200
1203    Park  85.7  88.7  90.3  87.3  88.000
1701    Yoon  76.8  80.2  83.5  75.4  78.975
1500    Choi  98.9  97.2  98.2  95.3  97.400
```

열 (과목)별 평균 계산, Avg 행(row) 추가

```
# pandas - df, calculation of average of each class

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
        'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
        'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
        'Math' : [95.2, 93.5, 90.3, 83.5, 98.2],
        'Sci': [75.9, 92.4, 87.3, 75.4, 95.3]
        }

df = pd.DataFrame(st_data, index=st_ids)
#print("df = \n", df)

avgs_per_student = df.mean(1)
   # mean with axes 1
print("\navgs_per_student =")
print(avgs_per_student)
df.loc[:, 'Avg'] = avgs_per_student

avgs_per_class = df.mean() # mean with axes 0
print("\navgs_per_class =")
print(avgs_per_class)

df.loc[len(df)]=avgs_per_class
df.at[len(df)-1, 'st_name'] = 'Total_Avg'
print("\ndf_with_avg =")
print(df)
```

```
avgs_per_student =
1201    89.775
2202    93.200
1203    88.000
1701    78.975
1500    97.400
dtype: float64

avgs_per_class =
Eng     89.90
Kor     90.58
Math    92.14
Sci     85.26
Avg     89.47
dtype: float64

df_with_avg =
         st_name   Eng    Kor   Math    Sci     Avg
1201         Kim  95.7  92.30  95.20  75.90  89.775
2202         Lee  92.4  94.50  93.50  92.40  93.200
1203        Park  85.7  88.70  90.30  87.30  88.000
1701        Yoon  76.8  80.20  83.50  75.40  78.975
1500        Choi  98.9  97.20  98.20  95.30  97.400
5     Total_Avg  89.9  90.58  92.14  85.26  89.470
```

데이터 프레임의 열을 구분하여 Series 생성

```
# pandas - indexing with name to obtain series

import pandas as pd

data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
        'st_id' : [1201, 2202, 1203, 1701, 2300],
        'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
        'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
        'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
        }

df = pd.DataFrame(data)

print("df = \n", df)
st_names = df['st_name']
print("st_names = df['st_name']")
print(st_names)

st_ids = df['st_id']
print("st_ids = df['st_id']")
print(st_ids)

eng_scores = df['Eng']
print("eng_scores = df['Eng']")
print(eng_scores)
```

```
df =
   st_name  st_id   Eng   Kor  Math
0     Kim   1201  95.7  92.3  95.2
1     Lee   2202  92.4  94.5  93.5
2    Park   1203  85.7  88.7  90.3
3    Yoon   1701  76.8  80.2  83.5
4    Choi   2300  98.9  97.2  98.2
st_names = df['st_name']
0     Kim
1     Lee
2    Park
3    Yoon
4    Choi
Name: st_name, dtype: object
st_ids = df['st_id']
0    1201
1    2202
2    1203
3    1701
4    2300
Name: st_id, dtype: int64
eng_scores = df['Eng']
0    95.7
1    92.4
2    85.7
3    76.8
4    98.9
Name: Eng, dtype: float64
```

데이터 정제 - dropna(), fillna(), isna()

```
# pandas - missing data handling - dropna(), fillna()

import pandas as pd
import numpy as np

st_ids = [1201, 2202, 1203, 1701, 2300]
data =\
{
   'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
   'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
   'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
   'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
}

df = pd.DataFrame(data, index = st_ids)
print("df = \n", df)
df['A'] = [1, np.nan, 1, 1, np.nan]
print("extended df = \n", df)

print("df.dropna() =")
print(df.dropna(how='any'))

print("df.fillna(value=0) = ")
print(df.fillna(value=0))

print("df.isna() = ")
print(df.isna())
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
2300    Choi  98.9  97.2  98.2
extended df =
      st_name   Eng   Kor  Math    A
1201     Kim  95.7  92.3  95.2  1.0
2202     Lee  92.4  94.5  93.5  NaN
1203    Park  85.7  88.7  90.3  1.0
1701    Yoon  76.8  80.2  83.5  1.0
2300    Choi  98.9  97.2  98.2  NaN
df.dropna() =
     st_name   Eng   Kor  Math    A
1201     Kim  95.7  92.3  95.2  1.0
1203    Park  85.7  88.7  90.3  1.0
1701    Yoon  76.8  80.2  83.5  1.0
df.fillna(value=0) =
     st_name   Eng   Kor  Math    A
1201     Kim  95.7  92.3  95.2  1.0
2202     Lee  92.4  94.5  93.5  0.0
1203    Park  85.7  88.7  90.3  1.0
1701    Yoon  76.8  80.2  83.5  1.0
2300    Choi  98.9  97.2  98.2  0.0
df.isna() =
      st_name    Eng    Kor   Math      A
1201    False  False  False  False  False
2202    False  False  False  False   True
1203    False  False  False  False  False
1701    False  False  False  False  False
2300    False  False  False  False   True
```

min(), max(), mean(), var(), std(), describe()

```
# pandas - min, max, mean, var, std, describe

import pandas as pd

data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
        'st_id' : [1201, 2202, 1203, 1701, 2300],
        'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
        'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
        'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
        }

df = pd.DataFrame(data)
print("df = \n", df)

eng_scores = df['Eng']
print("\neng_scores = df['Eng']")
print(eng_scores)

print("\neng_scores.min() = ", eng_scores.min())
print("eng_scores.max() = ", eng_scores.max())
print("eng_scores.mean() = ", eng_scores.mean())
print("eng_scores.var() = ", eng_scores.var())
print("eng_scores.std() = ", eng_scores.std())
print("\neng_scores.describe() = ", eng_scores.describe())
```

```
df =
   st_name  st_id   Eng   Kor  Math
0      Kim   1201  95.7  92.3  95.2
1      Lee   2202  92.4  94.5  93.5
2     Park   1203  85.7  88.7  90.3
3     Yoon   1701  76.8  80.2  83.5
4     Choi   2300  98.9  97.2  98.2

eng_scores = df['Eng']
0    95.7
1    92.4
2    85.7
3    76.8
4    98.9
Name: Eng, dtype: float64

eng_scores.min() =  76.8
eng_scores.max() =  98.9
eng_scores.mean() =  89.9
eng_scores.var() =  77.53500000000005
eng_scores.std() =  8.805396072863505

eng_scores.describe() =  count     5.000000
mean     89.900000
std       8.805396
min      76.800000
25%      85.700000
50%      92.400000
75%      95.700000
max      98.900000
Name: Eng, dtype: float64
```

데이터 분할, 병합

```
# pandas - DataFrame partitioning, concat()

import pandas as pd
import numpy as np

st_ids = [1201, 2202, 1203, 1701, 2300]
data =\
{
    'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
    'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
    'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
    'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
}

df = pd.DataFrame(data, index = st_ids)
print("df = \n", df)

df_partitions = [df[:2], df[2:4], df[4:]]
print("df_partitions[0] =")
print(df_partitions[0])
print("df_partitions[1] =")
print(df_partitions[1])
print("df_partitions[2] =")
print(df_partitions[2])

print("\npd.concat(df_partitions) =")
print(pd.concat(df_partitions))
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
2300    Choi  98.9  97.2  98.2
df_partitions[0] =
     st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
df_partitions[1] =
     st_name   Eng   Kor  Math
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
df_partitions[2] =
     st_name   Eng   Kor  Math
2300    Choi  98.9  97.2  98.2

pd.concat(df_partitions) =
     st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
2300    Choi  98.9  97.2  98.2
```

데이터 프레임의 결합 - merge(), join()

```
# pandas - DataFrame merge(), join()

import pandas as pd
import numpy as np

st_ids = [1201, 2202, 1203, 1701, 2300]
data_1 = {  'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'] }

df_1 = pd.DataFrame(data_1, index = st_ids)
print("df_1 = \n", df_1)

data_2 =\
{
   'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
   'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
   'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
}

df_2 = pd.DataFrame(data_2, index = st_ids)
print("df_2 = \n", df_2)

df = pd.merge(df_1, df_2, left_index=True, right_index=True, how='left')
print("df =\n", df)

print("\ndf_1.join(df_2, how='right') =")
print(df_1.join(df_2, how='right'))
```

```
df_1 =
      st_name
1201     Kim
2202     Lee
1203    Park
1701    Yoon
2300    Choi
df_2 =
       Eng   Kor  Math
1201  95.7  92.3  95.2
2202  92.4  94.5  93.5
1203  85.7  88.7  90.3
1701  76.8  80.2  83.5
2300  98.9  97.2  98.2
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
2300    Choi  98.9  97.2  98.2

df_1.join(df_2, how='right') =
     st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
2300    Choi  98.9  97.2  98.2
```

데이터프레임에 새로운 행을 추가 - append()

```
# pandas - calculate mean and add one more column with append()

import pandas as pd
st_ids = [1201, 2202, 1203, 1701, 2300]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
       'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
       'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
       'Math' : [95.2, 93.5, 90.3, 83.5, 98.2]
       }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

st_data_1 = {'st_name' : 'Hwang', 'Eng' : 95.0,  'Kor' : 85.7, 'Math' : 97.5}
df_1 = pd.DataFrame(st_data_1, index=[3000])
print("df_1 = ", df_1)

df_ext = df.append(df_1)
print("df_ext =", df_ext)
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
2300    Choi  98.9  97.2  98.2
df_1 =
    st_name   Eng   Kor  Math
3000   Hwang  95.0  85.7  97.5
df_ext =
     st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
2300    Choi  98.9  97.2  98.2
3000   Hwang  95.0  85.7  97.5
```

데이터 피봇팅 - pivot()

◆ pivot()

- stacked format으로 저장된 레코드 양식 (record format)의 데이터 프레임의 각 고윳값을 열(column)으로 표현

```
# pandas - pivoting with pivot()

import pandas as pd

print("Reading df_sensorXYZ from excel file")
df_sensorXYZ = \
   pd.read_excel("Sensor_Readings_XYZ.xlsx")

print("\nSensor_Readings_XYZ (in record format) = ")
print(df_sensorXYZ)

df_sensorXYZ_pivot = df_sensorXYZ.pivot(index='Time',
columns='variable', values='value')
print("\nSensor_Readings_XYZ after pivoting = ")
print(df_sensorXYZ_pivot)
```

Time	variable	value
2021-01-01 00:00:00	X	-1
2021-01-01 00:00:01	X	-3
2021-01-01 00:00:02	X	-5
2021-01-01 00:00:03	X	-2
2021-01-01 00:00:04	X	1
2021-01-01 00:00:05	X	2
2021-01-01 00:00:06	X	1
2021-01-01 00:00:07	X	0
2021-01-01 00:00:08	X	-1
2021-01-01 00:00:09	X	-3
2021-01-01 00:00:00	Y	1
2021-01-01 00:00:01	Y	1
2021-01-01 00:00:02	Y	2
2021-01-01 00:00:03	Y	3
2021-01-01 00:00:04	Y	6
2021-01-01 00:00:05	Y	5
2021-01-01 00:00:06	Y	6
2021-01-01 00:00:07	Y	6
2021-01-01 00:00:08	Y	4
2021-01-01 00:00:09	Y	1
2021-01-01 00:00:00	Z	3
2021-01-01 00:00:01	Z	2
2021-01-01 00:00:02	Z	5
2021-01-01 00:00:03	Z	7
2021-01-01 00:00:04	Z	10
2021-01-01 00:00:05	Z	8
2021-01-01 00:00:06	Z	9
2021-01-01 00:00:07	Z	7
2021-01-01 00:00:08	Z	5
2021-01-01 00:00:09	Z	3

```
Reading df_sensorXYZ from excel file

Sensor_Readings_XYZ (in record format) =
                   Time variable  value
0  2021-01-01 00:00:00.000000        X     -1
1  2021-01-01 00:00:01.000000        X     -3
2  2021-01-01 00:00:02.000000        X     -5
3  2021-01-01 00:00:03.000000        X     -2
4  2021-01-01 00:00:04.000000        X      1
5  2021-01-01 00:00:05.000000        X      2
6  2021-01-01 00:00:06.000001        X      1
7  2021-01-01 00:00:07.000000        X      0
8  2021-01-01 00:00:08.000000        X     -1
9  2021-01-01 00:00:09.000000        X     -3
10 2021-01-01 00:00:00.000000        Y      1
11 2021-01-01 00:00:01.000000        Y      1
12 2021-01-01 00:00:02.000000        Y      2
13 2021-01-01 00:00:03.000000        Y      3
14 2021-01-01 00:00:04.000000        Y      6
15 2021-01-01 00:00:05.000000        Y      5
16 2021-01-01 00:00:06.000001        Y      6
17 2021-01-01 00:00:07.000000        Y      6
18 2021-01-01 00:00:08.000000        Y      4
19 2021-01-01 00:00:09.000000        Y      1
20 2021-01-01 00:00:00.000000        Z      3
21 2021-01-01 00:00:01.000000        Z      2
22 2021-01-01 00:00:02.000000        Z      5
23 2021-01-01 00:00:03.000000        Z      7
24 2021-01-01 00:00:04.000000        Z     10
25 2021-01-01 00:00:05.000000        Z      8
26 2021-01-01 00:00:06.000001        Z      9
27 2021-01-01 00:00:07.000000        Z      7
28 2021-01-01 00:00:08.000000        Z      5
29 2021-01-01 00:00:09.000000        Z      3

Sensor_Readings_XYZ after pivoting =
variable                    X  Y   Z
Time
2021-01-01 00:00:00.000000 -1  1   3
2021-01-01 00:00:01.000000 -3  1   2
2021-01-01 00:00:02.000000 -5  2   5
2021-01-01 00:00:03.000000 -2  3   7
2021-01-01 00:00:04.000000  1  6  10
2021-01-01 00:00:05.000000  2  5   8
2021-01-01 00:00:06.000001  1  6   9
2021-01-01 00:00:07.000000  0  6   7
2021-01-01 00:00:08.000000 -1  4   5
2021-01-01 00:00:09.000000 -3  1   3
```

데이터 멜팅 - melt()

◆ melt()

- unpivoting으로 넓은 포멧을 긴 포멧으로 변환.
- 2개 이상의 열에서 레이블은 variable 열로, 데이터는 value 열로 이동 (unpivoting) 시켜 하나의 프레임을 재형성

```
# pandas - unpivoting with melt()

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 2300, 3000]
st_data =\
{
   'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi', 'Hwang'],
   'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9, 95.1],
   'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2, 85.7],
   'Math' : [95.2, 93.5, 90.3, 83.5, 98.2, 97.5]
}

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

melted_df = pd.melt(df, id_vars=['st_name'], value_vars=['Eng', 'Kor', 'Math'])
print("melted_df = \n", melted_df)
```

```
df =
      st_name   Eng   Kor  Math
1201     Kim  95.7  92.3  95.2
2202     Lee  92.4  94.5  93.5
1203    Park  85.7  88.7  90.3
1701    Yoon  76.8  80.2  83.5
2300    Choi  98.9  97.2  98.2
3000   Hwang  95.1  85.7  97.5
melted_df =
    st_name variable  value
0       Kim      Eng   95.7
1       Lee      Eng   92.4
2      Park      Eng   85.7
3      Yoon      Eng   76.8
4      Choi      Eng   98.9
5     Hwang      Eng   95.1
6       Kim      Kor   92.3
7       Lee      Kor   94.5
8      Park      Kor   88.7
9      Yoon      Kor   80.2
10     Choi      Kor   97.2
11    Hwang      Kor   85.7
12      Kim     Math   95.2
13      Lee     Math   93.5
14     Park     Math   90.3
15     Yoon     Math   83.5
16     Choi     Math   98.2
17    Hwang     Math   97.5
```

Excel 파일 출력

◆ openpyxl 설치

- > python -m pip install --upgrade openpyxl

```
C:\Users\Owner>python -m pip install --upgrade openpyxl
Collecting openpyxl
  Downloading openpyxl-3.0.5-py2.py3-none-any.whl (242 kB)
     |████████████████████████████████| 242 kB 2.2 MB/s
Collecting et-xmlfile
  Downloading et_xmlfile-1.0.1.tar.gz (8.4 kB)
Collecting jdcal
  Downloading jdcal-1.4.1-py2.py3-none-any.whl (9.5 kB)
Using legacy 'setup.py install' for et-xmlfile, since package 'wheel' is not installed.
Installing collected packages: jdcal, et-xmlfile, openpyxl
    Running setup.py install for et-xmlfile ... done
Successfully installed et-xmlfile-1.0.1 jdcal-1.4.1 openpyxl-3.0.5
```

```
. . . . .  # same as before

print("Writing df to excel file")
with pd.ExcelWriter("students_scores.xlsx") as excel_writer:
    df.to_excel(excel_writer, sheet_name='Students Records')
```

	st_name	st_id	Eng	Kor	Math	Avg
0	Kim	1201	95.7	92.3	95.2	94.4
1	Lee	2202	92.4	94.5	93.5	93.46667
2	Park	1203	85.7	88.7	90.3	88.23333
3	Yoon	1701	76.8	80.2	83.5	80.16667
4	Choi	2300	98.9	97.2	98.2	98.1
5	Avg	0	89.9	90.58	92.14	90.87333

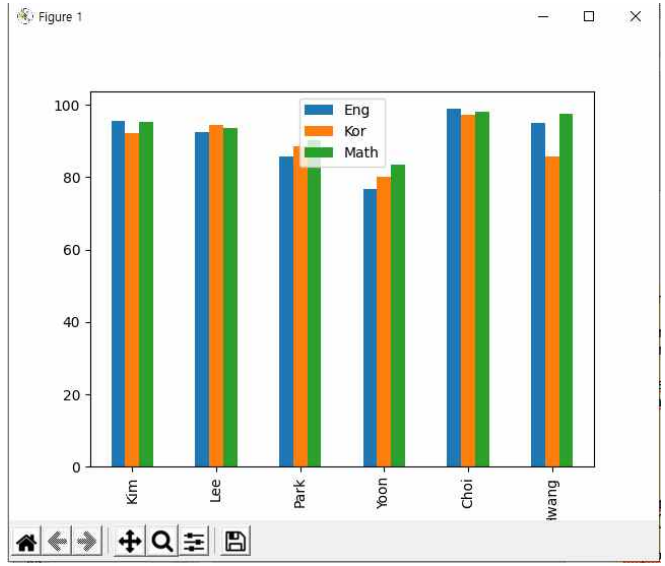
```
# pandas - df, calculation of average of each class, save to Excel

import pandas as pd

st_ids = [1201, 2202, 1203, 1701, 1500]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi'],
        'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9],
        'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2],
        'Math' : [95.2, 93.5, 90.3, 83.5, 98.2],
        'Sci': [75.9, 92.4, 87.3, 75.4, 95.3]
        }

df = pd.DataFrame(st_data, index=st_ids)
#print("df = \n", df)

avgs_per_student = df.mean(1) # mean with axes 1
print("\navgs_per_student =\n", avgs_per_student)

df.loc[:, 'Avg'] = avgs_per_student
#print("\ndf_with_avg =", df)

avgs_per_class = df.mean() # mean with axes 0
print("\navgs_per_class =\n", avgs_per_class)

df.loc[len(df)]=avgs_per_class
df.at[len(df)-1, 'st_name'] = 'Total_Avg'
print("\ndf_with_avg =\n", df)

print("Writing df to excel file")
with pd.ExcelWriter("students_scores.xlsx") as excel_writer:
    df.to_excel(excel_writer, sheet_name='Students Records')
```

```
avgs_per_student =
1201    89.775
2202    93.200
1203    88.000
1701    78.975
1500    97.400
dtype: float64

avgs_per_class =
Eng     89.90
Kor     90.58
Math    92.14
Sci     85.26
Avg     89.47
dtype: float64

df_with_avg =
       st_name   Eng    Kor   Math    Sci      Avg
1201       Kim  95.7  92.30  95.20  75.90  89.775
2202       Lee  92.4  94.50  93.50  92.40  93.200
1203      Park  85.7  88.70  90.30  87.30  88.000
1701      Yoon  76.8  80.20  83.50  75.40  78.975
1500      Choi  98.9  97.20  98.20  95.30  97.400
5     Total_Avg  89.9  90.58  92.14  85.26  89.470
Writing df to excel file
```

	A	B	C	D	E	F	G
1		st_name	Eng	Kor	Math	Sci	Avg
2	1201	Kim	95.7	92.3	95.2	75.9	89.775
3	2202	Lee	92.4	94.5	93.5	92.4	93.2
4	1203	Park	85.7	88.7	90.3	87.3	88
5	1701	Yoon	76.8	80.2	83.5	75.4	78.975
6	1500	Choi	98.9	97.2	98.2	95.3	97.4
7	5	Total_Avg	89.9	90.58	92.14	85.26	89.47

데이터 시각화 - plot()

```
# pandas - visualization with plot()

import pandas as pd
import matplotlib.pyplot as plt

st_ids = [1201, 2202, 1203, 1701, 2300, 3000]
st_data = {'st_name': ['Kim', 'Lee', 'Park', 'Yoon', 'Choi', 'Hwang'],
        'Eng' : [95.7, 92.4, 85.7, 76.8, 98.9, 95.1],
        'Kor' : [92.3, 94.5, 88.7, 80.2, 97.2, 85.7],
        'Math' : [95.2, 93.5, 90.3, 83.5, 98.2, 97.5]
        }

df = pd.DataFrame(st_data, index=st_ids)
print("df = \n", df)

df.plot(kind='bar', x='st_name', y=['Eng', 'Kor', 'Math'])
plt.show()
```



pandas 제공 method (1)

구분	Methods	설명
생성	Series()	pandas Series (시리즈) 생성
	DataFrame()	pandas DataFrame (데이터 프레임) 생성
파일 입출력	read_csv()	CSV(comma-separated values) 파일로 부터 데이터을 읽어 데이터 프레임 생성
	to_csv()	CSV(comma-separated values) 파일로 저장
	read_excel()	엑셀파일로 데이터을 읽어 데이터 프레임 생성
	to_excel()	엑셀파일로 저장
	read_hdf()	HDF5(Hierarchical Data Format version 5) 파일을 읽어 데이터 프레임 생성
	to_hdf()	HDF5(Hierarchical Data Format version 5) 파일로 데이터 프레임 출력
데이터 조합, 연관, 재형성	pivot()	데이터베이스나 액셀스프레드시트 등의 파일에서 스택양식(stacked format)/레코드양식(record format)으로 저장된 데이터을 읽어 만든 데이터 프레임에서 변수(variable)을 개별 칼럼(column)으로 표현하여 정규화시킴
	melt()	unpivoting으로 넓은 포멧을 긴 포멧으로 변환. 2개 이상의 열에서 레이블은 variable 열로, 데이터는 value 열로 이동 (unpivoting) 시켜 하나의 프레임을 재형성
	concat()	테이블의 행이나 열을 접합 (concatenation)
	merge()	공통 열이나 인덱스를 사용하여 데이터 병합
	join()	키 열이나 인덱스를 사용하여 데이터를 결합
	sort_values()	값에 따른 정렬
	sort_index()	인덱스에 따른 정렬
	append()	데이터 프레임에 행 추가
	drop()	행 삭제

pandas 제공 method (2)

구분	Methods	설명
서브셋 (행)	df.head(n)	데이터 프레임의 첫부분 n 항목 추출
	df.tail(n)	데이터 프레임의 끝부분 n 항목 추출
	df[i:j]	데이터 프레임의 i번째 행에서 j-1번째 행을 추출
	df[df.A>7]	데이터 프레임에서 지정된 조건 (예: A열의 값이 7보다 큰)에 만족된 항목만 추출
	df.loc[df["A"]<10, "B"]	조건을 만족하는 행(df["A"] < 10)에서 레이블("B")로 지정된 열을 추출
	df.drop_duplication()	중복된 항목의 삭제
	df.sample()	표본 추출
	df.nlargest()	제일 큰 n 항목
	df.nsmallest()	제일 작은 n 항목
서브셋 (열)	df['A']	데이터 프레임의 레이블이 A인 열을 추출
	df(['A', 'B', 'C'])	데이터 프레임의 레이블이 A, B, C인 열을 추출
	df.filter(regex='regex')	정규식 (regular expression)으로 표현된 조건에 따라 필터링
	df.loc[label] df.loc[index, label] df.at[index, label] df.loc[i:j, ['A', 'B']]	주어진 조건에 따른 명시적 레이블 검색/필터링 (레이블 기반, 행과 열의 이름 지정)
	df.iloc[i] df.iloc[i, j] df.iloc[i:j, n:m] df.iloc[[1,2,4], [3, 4]]	주어진 조건에 따른 명시적 포지션 검색/필터링 (인덱스 기반, 행과 열의 인덱스 지정)

pandas 제공 method (3)

구분	Methods	설명
데이터 분석	df['w'].value_counts()	각 카테고리에 대한 행의 수 계산
	dtypes	각 열의 데이터 유형 확인
	len(df)	데이터 프레임의 크기 (행의 개수)
	size()	열의 크기 (원소의 개수), 데이터 프레임에 포함된 원소의 개수 (행의 개수 x 열의 개수)
	df.describe()	데이터 프레임의 통계특성을 제공
	mean()	mean(): 각 열에 대한 평균값 계산 mean(1) : 각 행의 평균값 계산
	median()	중간값
	corr()	데이터 프레임의 열 사이의 상관관계 계산
	sum()	합
	count()	각 데이터프레임 열에서 null이 아닌 값의 발생 횟수
	quantile()	백분위수 (quantile) 산출
	apply(func, axis)	데이터 프레임에서 지정된 함수를 실행 (axis=0은 각 열에 대하여 실행, axis=1은 각 행에 대하여 실행)
	min()	지정된 series의 최솟값
	max()	지정된 series의 최댓값
	var()	분산 (variance)
	std()	표준편차 (standard deviation)

pandas 제공 method (4)

구분	Methods	설명
데이터 정제	dropna()	NaN (non-number) 항목을 삭제
	fillna(value=)	NaN (non-number) 항목을 value로 지정된 값으로 설정
	isna()	NaN (non-number)인지 확인
데이터 추가 및 분할	assign()	새 열(column)을 추가
	insert()	새 열(column)을 추가
	cut()	값들을 기반으로 이산화를 위한 구간별 나누기
	qcut()	지정된 분위수를 기반으로 구간별 나누기
데이터 그룹핑	groupby()	기준에 따라 몇 개의 그룹으로 데이터를 분할(그룹화)
멤버십	isin()	데이터 프레임에서 지정된 리스트의 값을 포함하는 지 확인
피봇 테이블	pivot_table(data, index, columns, values, aggfunc)	피봇 테이블의 생성
차트 작성	plot()	kind='line': 꺽은선 그래프 kind='bar': 막대그래프 kind='scatter': 산포도 그래프 kine='hist': 히스토그램

pandas 데이터 형식

데이터 형식	NumPy/pandas 객체	pandas 문자열 이름	비고
Boolean	np.bool	bool	단일 바이트로 저장
Integer	np.int	int	기본값은 64bit, unsinged int로 사용 가능 (np.uint)
Float	np.float	float	기본값은 64bit
Complex	mp.complex	complex	복소수
Object	np.object	O, object	객체, 일반적으로 문자열이며, 복수 개의 형식을 가진 열 또는 다른 파이썬 객체(tuple, list, dict 등)를 포함할 수 있음
Datetime	np.datetime64, pd.Timestamp	datetime64	nano-sec 단위의 정밀도를 가진 특정 시각
Timedelta	np.timedelta64 pd.Timedelta	timedelta64	nano-sec 단위의 시간 간격
Categroical	pd.Categorical	category	pandas에서만 사용

Pandas 시계열 주기

Date Offset	Frequency String	Description
DateOffset	None	Generic offset class, defaults to absolute 24 hours
BDay or BusinessDay	'B'	business day (weekday)
CDay or CustomBusinessDay	'C'	custom business day
Week	'W'	one week, optionally anchored on a day of the week
WeekOfMonth	'WOM'	the x-th day of the y-th week of each month
LastWeekOfMonth	'LWOM'	the x-th day of the last week of each month
MonthEnd	'M'	calendar month end
MonthBegin	'MS'	calendar month begin
BusinessMonthEnd	'BM'	business month end
BusinessMonthBegin	'BMS'	business month begin
CustomBusinessMonthEnd	'CBM'	custom business month end
CustomBusinessMonthBegin	'CBMS'	custom business month begin
QuarterEnd	'Q'	calendar quarter end
QuarterBegin	'QS'	calendar quarter begin
BQuarterEnd	'BQ	business quarter end
BQuarterBegin	'BQS'	business quarter begin
YearEnd	'A'	calendar year end
YearBegin	'AS' or 'BYS'	calendar year begin
BYearEnd	'BA'	business year end
BYearBegin	'BAS'	business year begin
BusinessHour	'BH'	business hour
CustomBusinessHour	'CBH'	custom business hour
Day	'D'	one absolute day
Hour	'H'	one hour
Minute	'T' or 'min'	one minute
Second	'S'	one second
Milli	'L' or 'ms'	one millisecond
Micro	'U' or 'us'	one microsecond
Nano	'N'	one nanosecond

시계열 데이터 분석의 예 – 최근 10년간 기온 분석 (1)

◆ 기상청 기온 측정 데이터 분석

- https://data.kma.go.kr
- 기후통계분석 → 통계분석 → 기온분석
- 자료구분 (일), 시작일자-종료일자, 지역/지점 설정
- 검색
- csv 다운로드

시계열 데이터 분석의 예 – 최근 10년간 기온 분석 (2)

◆ 대구 지역의 2000년 1월 1일 ~ 2020년 1월 27일 기온 측정 데이터

	날짜	지점	평균기온(℃)	최저기온(℃)	최고기온(℃)
2	2000-01-01	143	4.7	0	8.5
3	2000-01-02	143	6.5	3.1	11.5
4	2000-01-03	143	2.9	0	6.8
5	2000-01-04	143	2.3	-2.4	7.5
6	2000-01-05	143	4.9	-0.9	9.4
7	2000-01-06	143	6	1.2	9.6
8	2000-01-07	143	-1.7	-4	1.2
9	2000-01-08	143	-0.5	-5.3	4.1
10	2000-01-09	143	0	-1.6	1.9
11	2000-01-10	143	2.4	-1	6.8
12	2000-01-11	143	2.4	-3.7	7.2
13	2000-01-12	143	6.3	4.6	8.4
14	2000-01-13	143	4.9	0.9	7.6
15	2000-01-14	143	1.6	-0.1	5.6
16	2000-01-15	143	1.1	-3.9	6.2
17	2000-01-16	143	3.7	0.3	8.2
18	2000-01-17	143	2.4	-2.6	7.3
19	2000-01-18	143	3	0.6	6.8
20	2000-01-19	143	-1.7	-4.1	0.7
21	2000-01-20	143	-4.6	-6.7	-1.5
22	2000-01-21	143	-3.6	-7.8	1.5
23	2000-01-22	143	-1.6	-6.1	2
24	2000-01-23	143	2.7	-0.6	6.8
25	2000-01-24	143	2.8	0.8	5.7
26	2000-01-25	143	-1.5	-5.2	2.4
27	2000-01-26	143	-4.5	-7.3	-0.8
28	2000-01-27	143	-2.7	-7.2	3
29	2000-01-28	143	-0.7	-7.9	6.4
30	2000-01-29	143	-0.1	-5.5	4.9
31	2000-01-30	143	-0.4	-4.4	3.4
32	2000-01-31	143	-3.8	-7.1	0.2

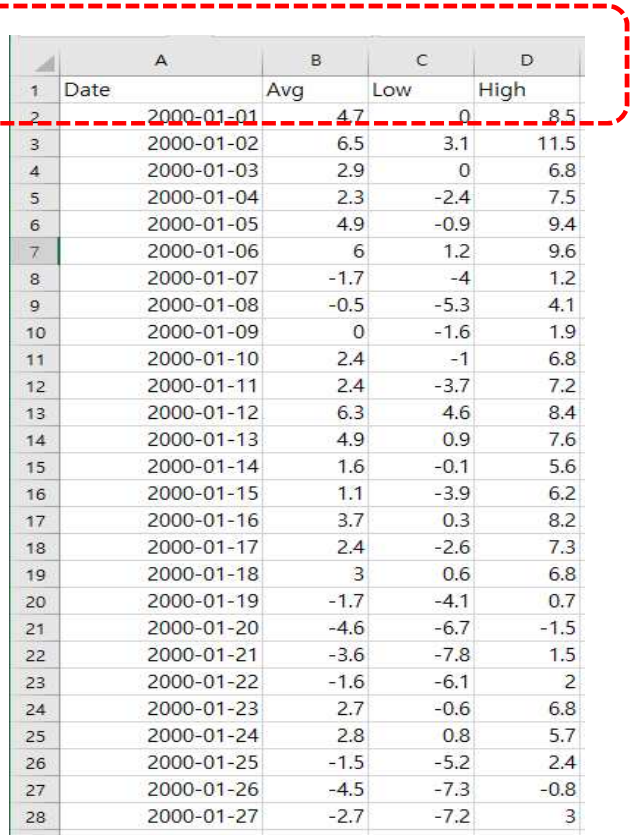
3662	2010-01-01	143	-3.3	-7.4	1.9
3663	2010-01-02	143	0.5	-5.4	7
3664	2010-01-03	143	-0.8	-3.9	3
3665	2010-01-04	143	-1	-3.1	1.5
3666	2010-01-05	143	-3.1	-6.4	-0.1
3667	2010-01-06	143	-5.4	-8.4	-0.7
3668	2010-01-07	143	-4.5	-7.5	-0.4
3669	2010-01-08	143	-2.8	-8.6	3.2
3670	2010-01-09	143	-0.1	-4.1	4.1
3671	2010-01-10	143	0.6	-2.8	3.3
3672	2010-01-11	143	0.2	-1.9	3.2
3673	2010-01-12	143	-3.2	-5.2	-0.2
3674	2010-01-13	143	-6.3	-8.9	-2.9
3675	2010-01-14	143	-4.3	-8.6	1.2
3676	2010-01-15	143	-1.2	-7	4.8
3677	2010-01-16	143	-0.7	-7.6	5.7
3678	2010-01-17	143	0.4	-7.2	7.4
3679	2010-01-18	143	1.8	-4.4	9.9
3680	2010-01-19	143	3.6	-3.5	10.9
3681	2010-01-20	143	9.1	6.4	13
3682	2010-01-21	143	4.6	-2.3	9.8
3683	2010-01-22	143	-2	-4.4	1.9
3684	2010-01-23	143	-1.5	-4.8	2.9
3685	2010-01-24	143	1.7	-3	7.4
3686	2010-01-25	143	2	-2.2	7.3
3687	2010-01-26	143	-0.5	-4.7	5.7
3688	2010-01-27	143	1.8	-4.2	6.3

7323	2020-01-10	143	2.2	-3.6	9.2
7324	2020-01-11	143	3.3	-0.2	8
7325	2020-01-12	143	2.7	0.2	6.5
7326	2020-01-13	143	2.1	-0.6	5.5
7327	2020-01-14	143	1.6	-0.9	5.8
7328	2020-01-15	143	0.9	-2.8	5.2
7329	2020-01-16	143	0.3	-4.7	7.2
7330	2020-01-17	143	1.4	-2.1	8
7331	2020-01-18	143	3.1	-1.1	8.7
7332	2020-01-19	143	3.2	-3.2	9
7333	2020-01-20	143	4.6	2.4	8.1
7334	2020-01-21	143	1.9	-2	7.9
7335	2020-01-22	143	2.9	0.2	5.7
7336	2020-01-23	143	5	2.7	8.6
7337	2020-01-24	143	6.1	0.5	11.8
7338	2020-01-25	143	7.2	5.4	9.8
7339	2020-01-26	143	6.7	1.7	11.2
7340	2020-01-27	143	6.2	4.5	8.5

CSV 데이터 읽기 – pandas.read_csv()

```
# pandas - handling CSV data

import pandas as pd

Temp_DG = pd.read_csv("ta_20210113.csv")
print(" Temp_DG = \n", Temp_DG)
```

```
Temp_DG =
            Date   Avg   Low  High
0     2000-01-01   4.7   0.0   8.5
1     2000-01-02   6.5   3.1  11.5
2     2000-01-03   2.9   0.0   6.8
3     2000-01-04   2.3  -2.4   7.5
4     2000-01-05   4.9  -0.9   9.4
...          ...   ...   ...   ...
7678  2021-01-08 -10.4 -13.6  -5.8
7679  2021-01-09  -8.0 -11.4  -3.1
7680  2021-01-10  -4.7 -10.8   1.2
7681  2021-01-11  -4.2  -8.5  -0.8
7682  2021-01-12  -1.5  -8.8   4.3

[7683 rows x 4 columns]
```



	A	B	C	D
1	Date	Avg	Low	High
2	2000-01-01	4.7	0	8.5
3	2000-01-02	6.5	3.1	11.5
4	2000-01-03	2.9	0	6.8
5	2000-01-04	2.3	-2.4	7.5
6	2000-01-05	4.9	-0.9	9.4
7	2000-01-06	6	1.2	9.6
8	2000-01-07	-1.7	-4	1.2
9	2000-01-08	-0.5	-5.3	4.1
10	2000-01-09	0	-1.6	1.9
11	2000-01-10	2.4	-1	6.8
12	2000-01-11	2.4	-3.7	7.2
13	2000-01-12	6.3	4.6	8.4
14	2000-01-13	4.9	0.9	7.6
15	2000-01-14	1.6	-0.1	5.6
16	2000-01-15	1.1	-3.9	6.2
17	2000-01-16	3.7	0.3	8.2
18	2000-01-17	2.4	-2.6	7.3
19	2000-01-18	3	0.6	6.8
20	2000-01-19	-1.7	-4.1	0.7
21	2000-01-20	-4.6	-6.7	-1.5
22	2000-01-21	-3.6	-7.8	1.5
23	2000-01-22	-1.6	-6.1	2
24	2000-01-23	2.7	-0.6	6.8
25	2000-01-24	2.8	0.8	5.7
26	2000-01-25	-1.5	-5.2	2.4
27	2000-01-26	-4.5	-7.3	-0.8
28	2000-01-27	-2.7	-7.2	3

시계열 데이터 파일 읽기 및 분석

```
# pandas - handling CSV data

import pandas as pd

Temp_DG = pd.read_csv("ta_20210113.csv")
print("Temp_DG = \n", Temp_DG)
#Avg_temp_DG = Temp_DG['Avg']
#print("Avg_Temp_DG =\n", Avg_temp_DG)

print("Temp_DG.describe() =")
print(Temp_DG.describe())
temp_DG_highest = Temp_DG['High'].max()
print("temp_DG_highest = ", temp_DG_highest)
temp_DG_lowest = Temp_DG['Low'].min()
print("temp_DG_lowest = ", temp_DG_lowest)

Temp_DG_highest_day = Temp_DG[Temp_DG.High >=
temp_DG_highest]
print("Temp_DG_highest_day =\n", Temp_DG_highest_day)

Temp_DG_lowest_day = Temp_DG[Temp_DG.Low <=
temp_DG_lowest]
print("Temp_DG_lowest_day =\n", Temp_DG_lowest_day)
```

```
Temp_DG =
            Date   Avg   Low  High
0     2000-01-01   4.7   0.0   8.5
1     2000-01-02   6.5   3.1  11.5
2     2000-01-03   2.9   0.0   6.8
3     2000-01-04   2.3  -2.4   7.5
4     2000-01-05   4.9  -0.9   9.4
...          ...   ...   ...   ...
7678  2021-01-08 -10.4 -13.6  -5.8
7679  2021-01-09  -8.0 -11.4  -3.1
7680  2021-01-10  -4.7 -10.8   1.2
7681  2021-01-11  -4.2  -8.5  -0.8
7682  2021-01-12  -1.5  -8.8   4.3

[7683 rows x 4 columns]
Temp_DG.describe() =
                Avg           Low          High
count  7681.000000  7683.000000  7682.000000
mean     14.539188    10.013276    19.774863
std       9.533006     9.726442     9.767435
min     -10.400000   -13.900000    -7.600000
25%       6.200000     1.400000    11.400000
50%      15.400000    10.400000    21.100000
75%      22.700000    18.700000    27.900000
max      33.100000    28.600000    39.200000
temp_DG_highest =  39.2
temp_DG_lowest =  -13.9
Temp_DG_highest_day =
            Date   Avg   Low  High
6782  2018-07-27  32.4  28.6  39.2
Temp_DG_lowest_day =
            Date  Avg   Low  High
6601  2018-01-27 -5.6 -13.9   2.8
```

Matplotlib

Matplotlib 패키지

◆ Matplotlib 패키지

- Matplotlib 패키지는 문서 및 서적 출판에서 사용 가능한 품질의 2차원 그래프 및 도형 출력 라이브러리이며 전문 인쇄용 출력 포맷을 지원
- Matplotlib는 파이썬 스크립트, IPython 쉘, Jupyter 노트북, 웹 응용 서버 등에서 사용할 수 있으며, 다수의 그래픽 사용자 접속 툴킷 (tool kit)에서 사용 가능
- 파이썬 프로그래밍과 함께 Matplotlib를 사용하면 계산 및 연산 결과들을 2차원 평면상에 그래프로 표시할 수 있어 시각적으로 쉽게 이해할 수 있게 함
- 특히 주어진 데이터에 대한 통계 분석 결과나 최소 오차의 근사방정식 등을 데이터와 함께 표시함으로써 이해를 돕고, 구현된 알고리즘에 따라 어떤 차이가 나는가를 쉽게 이해할 수 있게 함

Matplotlib.pyplot

◆ Matplotlib.pyplot의 함수 및 그래프 종류

Matplotlib.pyplot 함수	설명
plot()	선 그래프
bar()	막대그래프
hist()	돗수 분포 그래프 및 분포도 계산
scatter()	2차원 및 3차원 산포도 그래프
plot_wireframe()	3차원 와이어 프레임 그래프
plot_surface()	3차원 곡면 그래프
contour()	3차원 등고선 그래프

◆ Matplotlib.pyplot plot() 함수

Matplotlib.pyplot 함수	설명
plot()	plot(y) : 배열 y와 x=[0, 1, ..., len(y)-1], 기본 선 스타일, 기본 색상
	plot(x, y) : 배열 x, y, 기본 선 스타일, 기본 색상
	plot(y, fmt) : 배열 y와 x=[0, 1, ..., len(y)-1], 포맷 문자열 fmt
	plot(x, y, fmt) : 배열 x, y, 포맷 문자열 fmt
	plot(x1, y1, fmt1, x2, y2, fmt2, ...) : (배열 x1, y1, 포맷 문자열 fmt1), (배열 x2, y2, 포맷 문자열 fmt2), ...

Matplotlib의 마커 및 속성 설정

◆ Matplotlib의 주요 마커

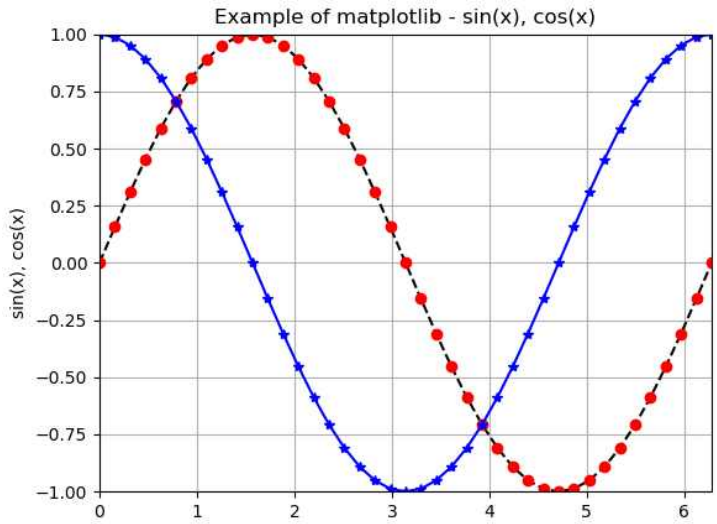
marker	설명	marker	설명
'o'	circle	'+'	plus
'v', '^', '<', '>'	triangle_down, up, left, right	'x'	x
'8'	octagon	'D'	diamond
's'	square	'd'	thin_diamond
'p'	pentagon	'|'	vertical line (vline)
'*'	star (asterisk)	'_'	horizontal line (hline)

◆ Matplotlib의 색상 단축 문자

색상 단축 문자	설명	색상 단축 문자	설명
'b'	blue	'k'	black
'g'	green	'm'	magenta
'r'	red	'y'	yellow
'c'	cyan	'w'	white

Example of sin(x), cos(x)

```
# matplotlib(3) - sin(x), cos(x)

import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2*np.pi, num=41)
sin_x, cos_x = np.sin(x), np.cos(x)
plt.plot(x, sin_x, "k--", x, sin_x, "ro")
plt.plot(x, cos_x, "b-", x, cos_x, "b*")

xmin, xmax, ymin, ymax = x[0], x[-1], -1, 1
plt.axis([xmin, xmax, ymin, ymax])
plt.xlabel("x")
plt.ylabel("sin(x), cos(x)")
plt.title("Example of matplotlib - sin(x), cos(x)")
plt.grid(True)
plt.savefig("matplot_003_sin_cos.png")
plt.show()
```

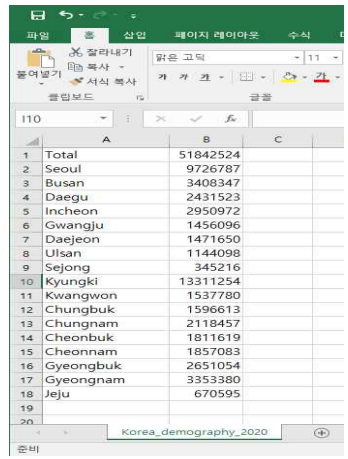


Matplotlib.pyplot – bar()

```
# Bar chart of Korean demography 2020
import matplotlib.pyplot as plt
import csv # comma separated value

plt.style.use('ggplot')
data_file_name = "Korea_demography_2020.csv"
f = open(data_file_name)
demo_data = csv.reader(f)
total = next(demo_data)
demography = []
city_names = [] # city names
for row in demo_data:
    city_name = row[0]
    city_names.append(city_name)
    demo_str = row[1].replace(',', '')
    demography.append(int(demo_str))
x_pos = [i for i, dmg in enumerate(demography)]
print("Demography of Korea in 2020: ")
for i in range(len(city_names)):
    print("({:10s}: {:10d}), ".format(city_names[i], demography[i]))

plt.bar(x_pos, demography, color='green')
plt.xlabel("City / Region in Korea")
plt.ylabel("Per-city/region Demography")
plt.xticks(x_pos, city_names, fontsize=7, rotation=30)
plt.title("Demography of Korea in 2020")
plt.show()
```

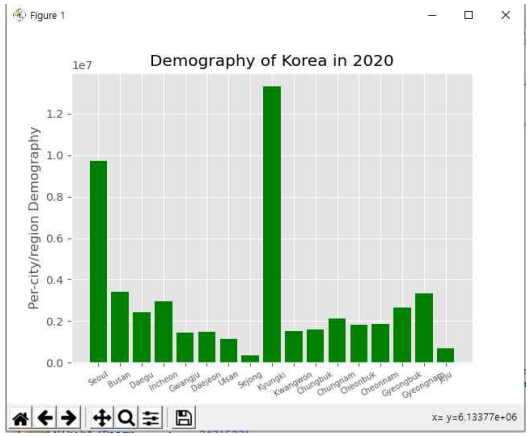
2020년 9월 기준

순위	광역자치단체	인구	비율
1위	경기도	13,388,485명	25.41%
2위	서울특별시	9,699,232명	18.81%
3위	부산광역시	3,399,749명	6.61%
4위	경상남도	3,343,770명	6.49%
5위	인천광역시	2,942,553명	5.70%
6위	경상북도	2,640,003명	5.15%
7위	대구광역시	2,426,849명	4.72%
8위	충청남도	2,120,559명	4.10%
9위	전라남도	1,851,124명	3.60%
10위	전라북도	1,806,441명	3.52%
11위	충청북도	1,598,536명	3.08%
12위	강원도	1,541,104명	2.97%
13위	대전광역시	1,469,099명	2.86%
14위	광주광역시	1,453,952명	2.82%
15위	울산광역시	1,139,368명	2.22%
16위	제주특별자치도	672,948명	1.29%
17위	세종특별자치시	348,014명	0.64%
총합		51,841,786명	100%



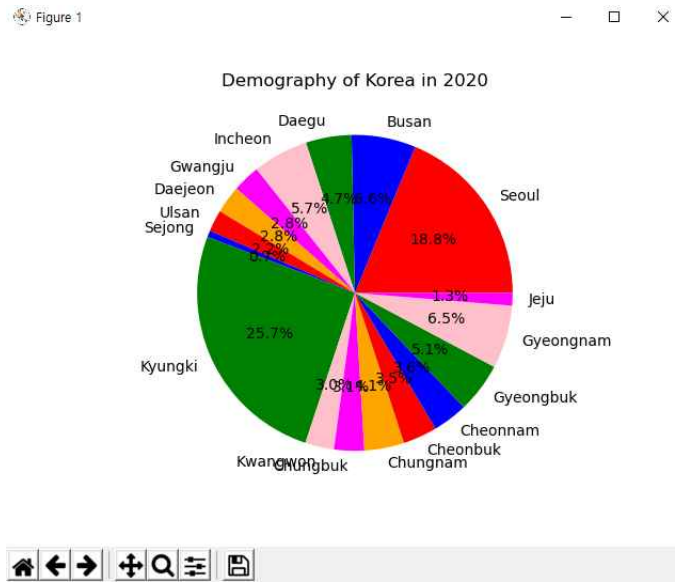
1	Total	51842524
2	Seoul	9726787
3	Busan	3408347
4	Daegu	2431523
5	Incheon	2950972
6	Gwangju	1456096
7	Daejeon	1471650
8	Ulsan	1144098
9	Sejong	345216
10	Kyungki	13311254
11	Kwangwon	1537780
12	Chungbuk	1596613
13	Chungnam	2118457
14	Cheonbuk	1811619
15	Cheonnam	1857083
16	Gyeongbuk	2651054
17	Gyeongnam	3353380
18	Jeju	670595

```
Demography of Korea in 2020:
(Seoul      :    9726787),
(Busan      :    3408347),
(Daegu      :    2431523),
(Incheon    :    2950972),
(Gwangju    :    1456096),
(Daejeon    :    1471650),
(Ulsan      :    1144098),
(Sejong     :     345216),
(Kyungki    :   13311254),
(Kwangwon   :    1537780),
(Chungbuk   :    1596613),
(Chungnam   :    2118457),
(Cheonbuk   :    1811619),
(Cheonnam   :    1857083),
(Gyeongbuk  :    2651054),
(Gyeongnam  :    3353380),
(Jeju       :     670595),
```



Matplotlib.pyplot – pie()

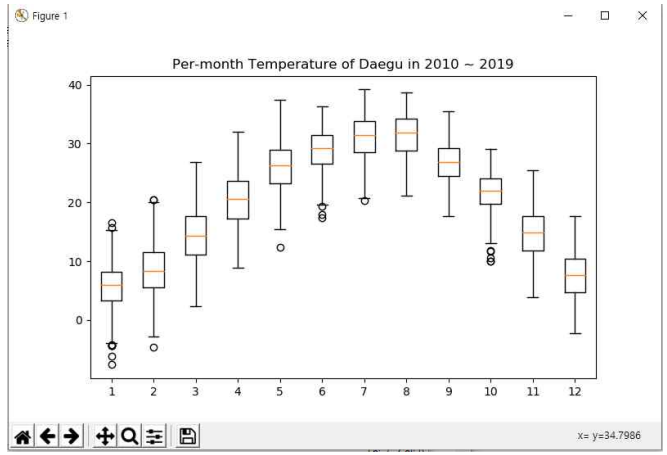
```
# Pie chart of Korean demography 2020
import matplotlib.pyplot as plt
import csv

data_file_name = "Korea_demography_2020.csv"
f = open(data_file_name)
demo_data = csv.reader(f)
total = next(demo_data)
demography = []
city_names = []
for row in demo_data:
    city_name = row[0]
    city_names.append(city_name)
    demo_str = row[1].replace(',', '')
    demography.append(int(demo_str))

print("Demography of Korea in 2020: ", demography)
color = ['red', 'blue', 'green', 'pink', 'magenta', 'orange']
plt.axis('equal')
plt.pie(demography, labels=city_names, autopct='%.1f%%', colors=color)
plt.title("Demography of Korea in 2020")
#plt.legend(loc='best')
plt.show()
```



Matplotlib.pyplot – boxplot()

```
# Time series data analysis - boxplot of per-month temperature
import matplotlib.pyplot as plt
import csv

data_file_name = "daegu_temp_20010101_20200127.csv"
f = open(data_file_name)
temp_data = csv.reader(f)
header = next(temp_data)
temp_month = [[], [], [], [], [], [], [], [], [], [], [], []]
for daily_temp in temp_data:
    if daily_temp[4] != '':
        m = int(daily_temp[0].split('-')[1])-1 # month index 0 ~ 11
        temp_month[m].append(float(daily_temp[4]))

plt.boxplot(temp_month)
plt.title("Per-month Temperature of Daegu in 2010 ~ 2019")
plt.show()
```

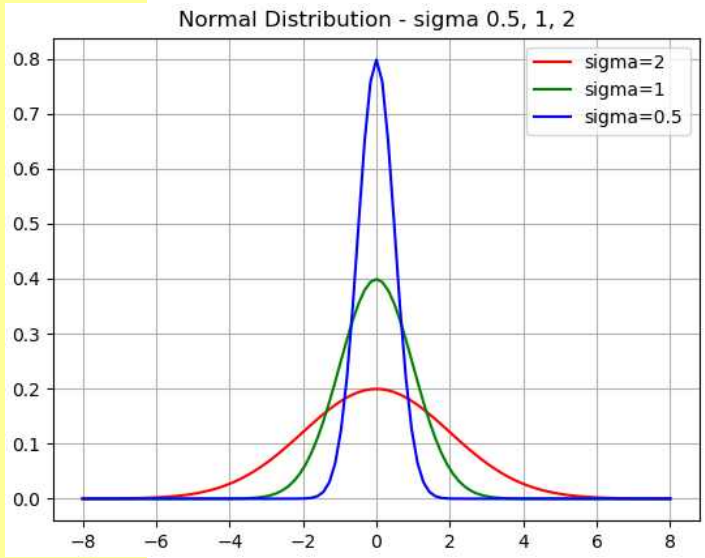
1	날짜	지점	평균기온(℃	최저기온(℃)	최고기온(℃)
2	2000-01-01	143	4.7	0	8.5
3	2000-01-02	143	6.5	3.1	11.5



Example of Normal (Gaussian) Distribution

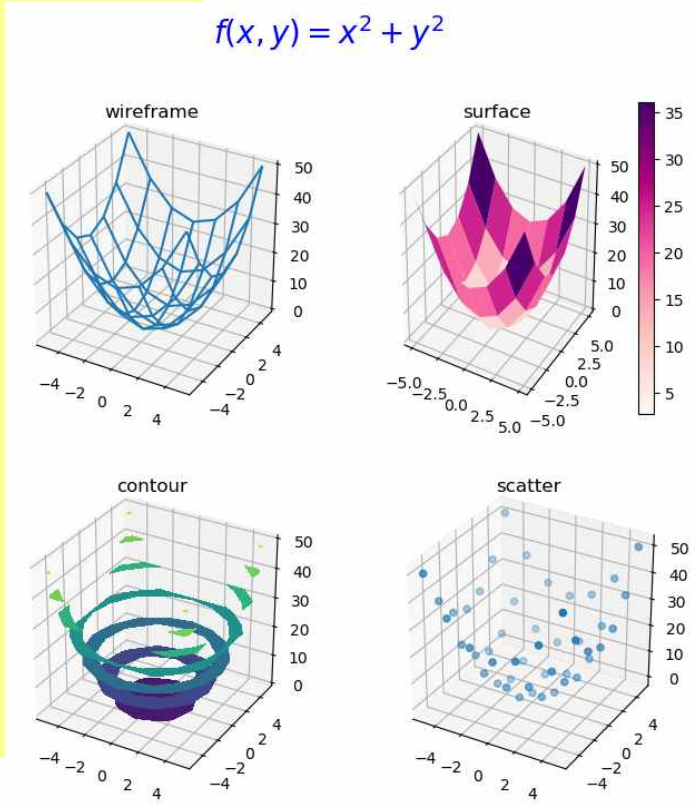
$$y = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right]$$

```
# matplotlib(4) - Normal, Gaussian Distribution

import numpy as np
import matplotlib.pyplot as plt

def gauss(mu, sigma, x):
    y = 1.0/(sigma*np.sqrt(2*np.pi))*np.exp(-((x - mu)**2)/(2*sigma**2))
    return y

mu, sigma = 0, 2
x = np.linspace(-4*sigma, 4*sigma, num=101)
y1 = gauss(mu, sigma, x)
plt.plot(x, y1, color="red", label="sigma=2")

mu, sigma = 0, 1
y2 = gauss(mu, sigma, x)
plt.plot(x, y2, color="green", label="sigma=1")

mu, sigma = 0, 0.5
y3 = gauss(mu, sigma, x)
plt.plot(x, y3, color="blue", label="sigma=0.5")

plt.title("Normal Distribution - sigma 0.5, 1, 2")

plt.legend(loc="best")
plt.grid(True)
plt.savefig("matplot_004_GaussDist.png")
plt.show()
```



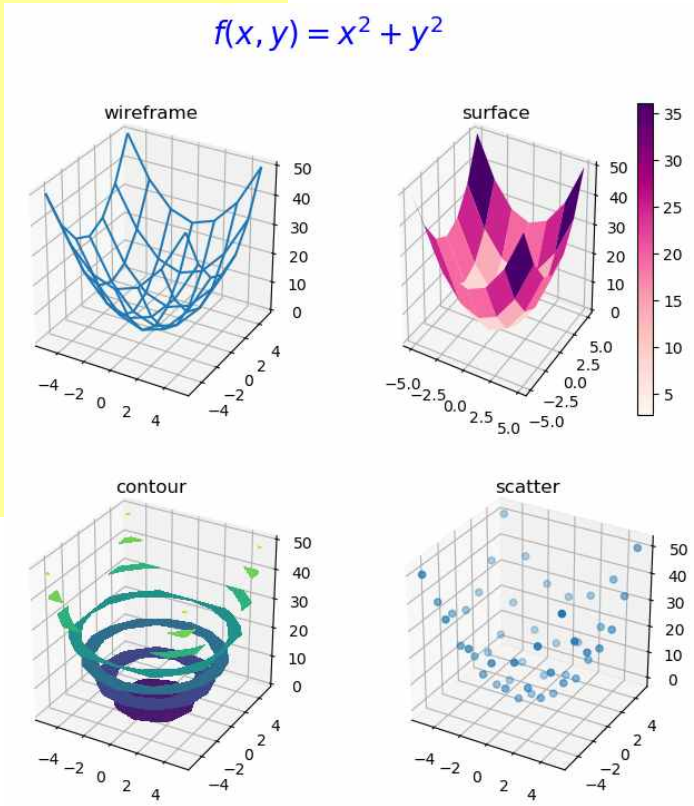
3차원 그래프 예제 (1)

```
# matplotlib - 3D graphs (1)

import numpy as np
import matplotlib.pyplot as plt

from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(8, 8))
fig.suptitle("$f(x,y) = x^2+y^2$", color='b', fontsize=20)

x = np.linspace(-5, 5, 7)
y = np.linspace(-5, 5, 7)
X, Y = np.meshgrid(x, y)
Z = X**2 + Y**2

ax1 = fig.add_subplot(221, projection='3d')
surf = ax1.plot_wireframe(X, Y, Z)
ax1.set_title("wireframe")

ax2 = fig.add_subplot(222, projection='3d')
surf = ax2.plot_surface(X, Y, Z, rstride=1,  cstride=1, cmap=cm.RdPu)
fig.colorbar(surf)
ax2.set_title("surface")
```



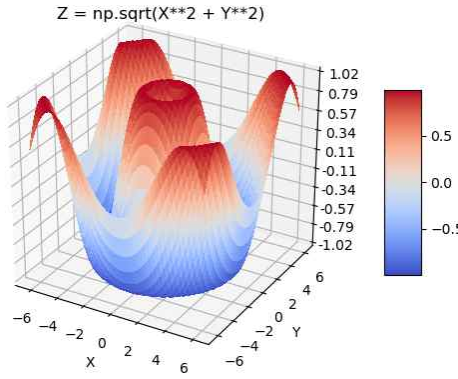
3차원 그래프 예제 (1)

```
# matplotlib - 3D graphs (2)

ax3 = fig.add_subplot(223, projection='3d')
#surf = ax3.contour(X, Y, Z) # countour lines
surf = ax3.contourf(X, Y, Z) #filled contours
ax3.set_title("contour")

ax4 = fig.add_subplot(224, projection='3d')
surf = ax4.scatter(X, Y, Z)
ax4.set_title("scatter")

plt.grid(True)
plt.savefig("matplot_005_3D graphic.png", bbox_inches='tight')
plt.show()
```



3차원 그래프 예제 (2)

```
# matplotlib - 3D graphic

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
PI = np.pi
x = np.arange(-2*PI, 2*PI, 0.25)
y = np.arange(-2*PI, 2*PI, 0.25)
X, Y = np.meshgrid(x, y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

print("X = \n", X)
print("Y = \n", Y)
print("Z = \n", Z)

ax = fig.gca(projection = '3d')
surf = ax.plot_surface(X, Y, Z, rstride=1,\
 cstride=1, cmap=cm.coolwarm, linewidth=0, antialiased=False)
ax.set_zlim(-1.02, 1.02)
plt.xlabel('X')
plt.ylabel('Y')
plt.title("Z = np.sqrt(X**2 + Y**2)")

ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.show()
```

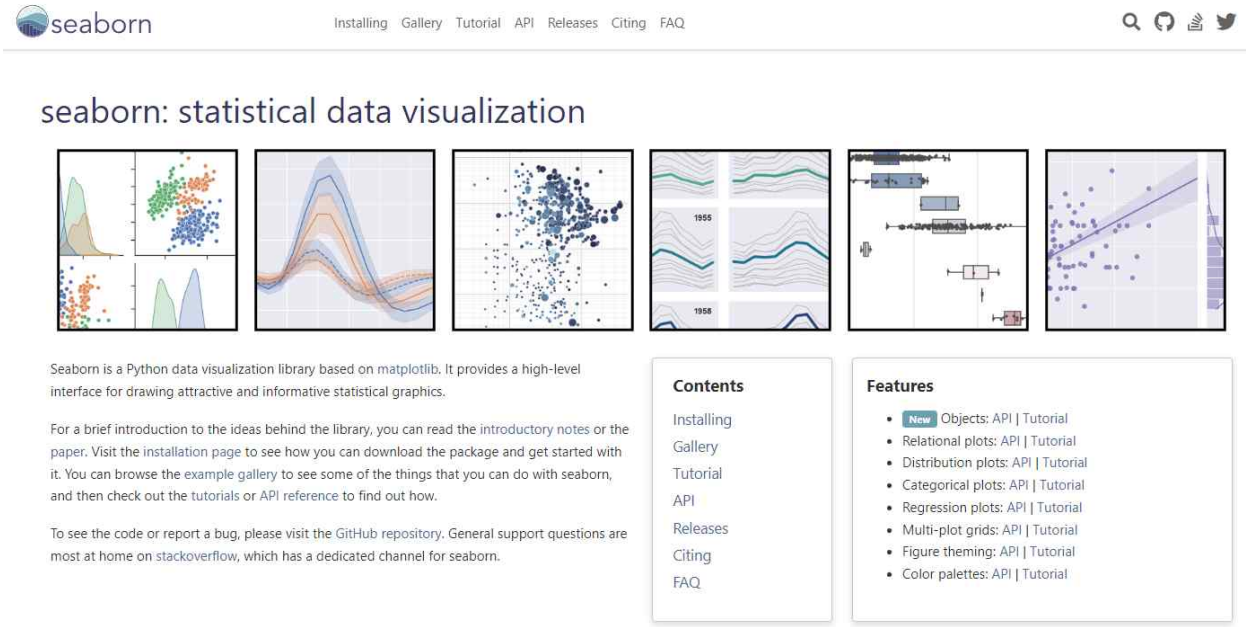


Seaborn

Seaborn 확장 모듈

◆ Seaborn 확장 모듈 이란?

- Matplotlib 확장 모듈에 다양한 색상 테마와 다양한 차트 기능을 추가
- 다양한 연습용 데이터 세트 포함
- https://seaborn.pydata.org/index.html



Seaborn 확장 모듈

◆ Seaborn 확장 모듈 설치 - >python -m pip install --upgrade seaborn

```
C:₩Users₩Owner>python -m pip install --upgrade pip
Requirement already satisfied: pip in c:₩users₩owner₩appdata₩local₩programs₩python₩python311₩lib₩site-packages (22.3.1)

C:₩Users₩Owner>python -m pip install --upgrade seaborn
Collecting seaborn
  Downloading seaborn-0.12.2-py3-none-any.whl (293 kB)
     ---------------------------------------- 293.3/293.3 kB 9.1 MB/s eta 0:00:00
Collecting numpy!=1.24.0,>=1.17
  Downloading numpy-1.24.1-cp311-cp311-win_amd64.whl (14.8 MB)
     ---------------------------------------- 14.8/14.8 MB 11.5 MB/s eta 0:00:00
Collecting pandas>=0.25
  Downloading pandas-1.5.3-cp311-cp311-win_amd64.whl (10.3 MB)
     ---------------------------------------- 10.3/10.3 MB 11.3 MB/s eta 0:00:00
Collecting matplotlib!=3.6.1,>=3.1
  Downloading matplotlib-3.6.3-cp311-cp311-win_amd64.whl (7.2 MB)
     ---------------------------------------- 7.2/7.2 MB 11.5 MB/s eta 0:00:00
Collecting contourpy>=1.0.1
  Downloading contourpy-1.0.7-cp311-cp311-win_amd64.whl (162 kB)
     ---------------------------------------- 163.0/163.0 kB 9.5 MB/s eta 0:00:00
Collecting cycler>=0.10
  Using cached cycler-0.11.0-py3-none-any.whl (6.4 kB)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.38.0-py3-none-any.whl (965 kB)
     ---------------------------------------- 965.4/965.4 kB 12.1 MB/s eta 0:00:00
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.4-cp311-cp311-win_amd64.whl (55 kB)
     ---------------------------------------- 55.4/55.4 kB ? eta 0:00:00
Collecting packaging>=20.0
  Downloading packaging-23.0-py3-none-any.whl (42 kB)
     ---------------------------------------- 42.7/42.7 kB 2.0 MB/s eta 0:00:00
Collecting pillow>=6.2.0
  Downloading Pillow-9.4.0-cp311-cp311-win_amd64.whl (2.5 MB)
     ---------------------------------------- 2.5/2.5 MB 12.1 MB/s eta 0:00:00
Collecting pyparsing>=2.2.1
  Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
     ---------------------------------------- 98.3/98.3 kB 5.5 MB/s eta 0:00:00
Collecting python-dateutil>=2.7
  Using cached python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
Collecting pytz>=2020.1
  Downloading pytz-2022.7.1-py2.py3-none-any.whl (499 kB)
     ---------------------------------------- 499.4/499.4 kB 15.8 MB/s eta 0:00:00
Collecting six>=1.5
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pytz, six, pyparsing, pillow, packaging, numpy, kiwisolver, fonttools, cycler, python-dateutil, contourpy, pa
ndas, matplotlib, seaborn
Successfully installed contourpy-1.0.7 cycler-0.11.0 fonttools-4.38.0 kiwisolver-1.4.4 matplotlib-3.6.3 numpy-1.24.1 packaging-23.0 pandas-1
.5.3 pillow-9.4.0 pyparsing-3.0.9 python-dateutil-2.8.2 pytz-2022.7.1 seaborn-0.12.2 six-1.16.0
```

Seaborn 확장 모듈

◆ Seaborn의 연습용 데이터 셋

```
# Seaborn-based data visualization

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

print("Seaborn dataset : ", sns.get_dataset_names())

# load data set as pandas data frame
df_titanic = sns.load_dataset("titanic")
print("\nSeaborn titanic :\n", df_titanic)
```

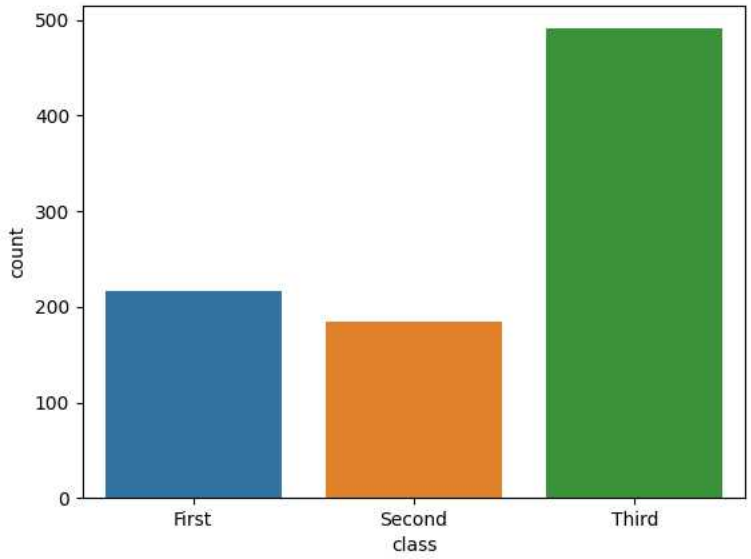
```
Seaborn dataset :  ['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'di
amonds', 'dots', 'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue', 'healthexp', 'i
ris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic']

Seaborn titanic :
     survived  pclass     sex   age  ...  deck  embark_town  alive  alone
0           0       3    male  22.0  ...   NaN  Southampton     no  False
1           1       1  female  38.0  ...     C    Cherbourg    yes  False
2           1       3  female  26.0  ...   NaN  Southampton    yes   True
3           1       1  female  35.0  ...     C  Southampton    yes  False
4           0       3    male  35.0  ...   NaN  Southampton     no   True
..        ...     ...     ...   ...  ...   ...          ...    ...    ...
886         0       2    male  27.0  ...   NaN  Southampton     no   True
887         1       1  female  19.0  ...     B  Southampton    yes   True
888         0       3  female   NaN  ...   NaN  Southampton     no  False
889         1       1    male  26.0  ...     C    Cherbourg    yes   True
890         0       3    male  32.0  ...   NaN   Queenstown     no   True

[891 rows x 15 columns]
```

(참고자료: **9 Seaborn Datasets for Data Science _ ML Beginners**
https://python.plainenglish.io/9-datasets-for-data-science-ml-beginners-cfb57df53fda)

Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn countplot()

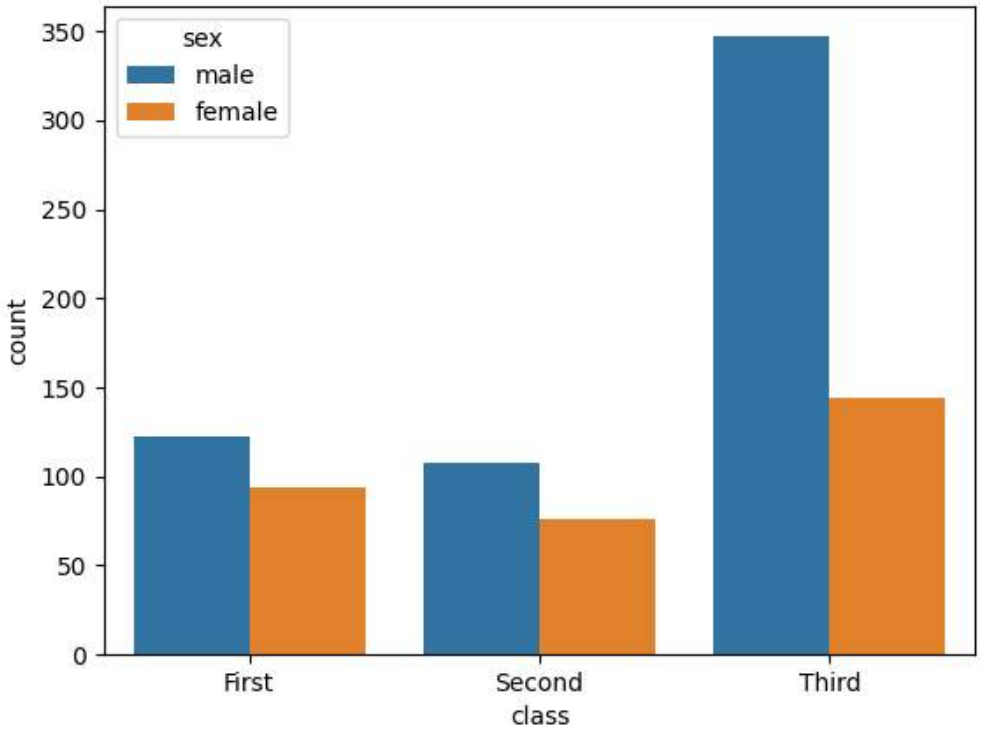
```
# Seaborn-based data visualization

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_titanic = sns.load_dataset("titanic")
print("\nSeaborn titanic :\n", df_titanic)

sns.countplot(x='class', data=df_titanic)
plt.show()
```



Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn countplot()

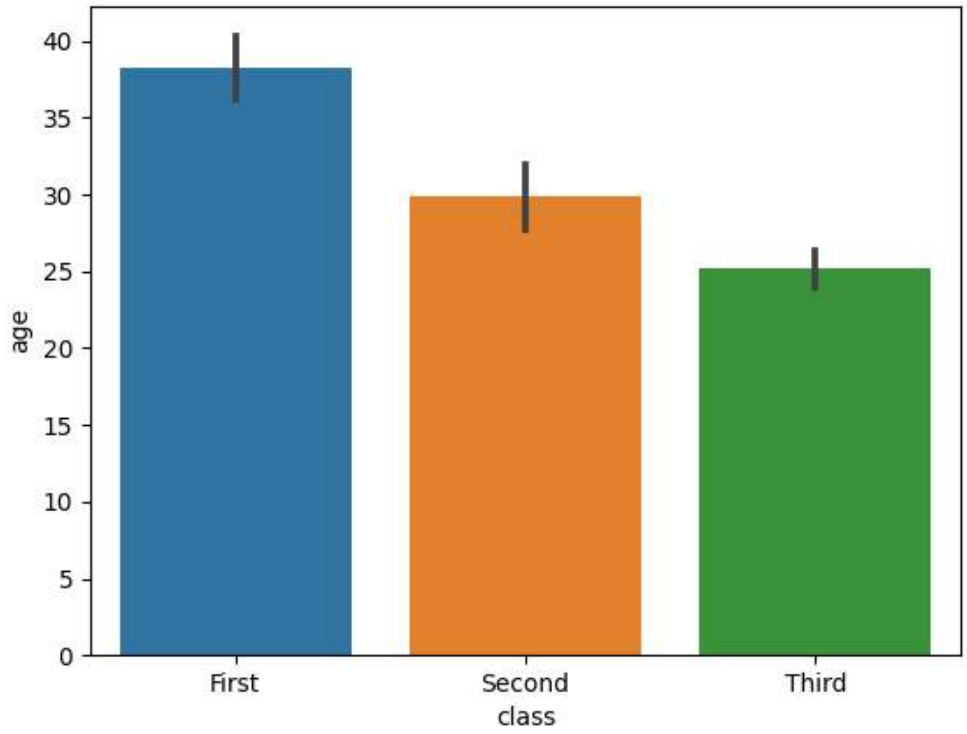
```
# Seaborn-based data visualization

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

print("Seaborn dataset : ", sns.get_dataset_names())

# load as Pandas data frame
df_titanic = sns.load_dataset("titanic")
print("\nSeaborn titanic :\n", df_titanic)

sns.countplot(x='class', hue='sex', data=df_titanic)
plt.show()
```



Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn barplot()

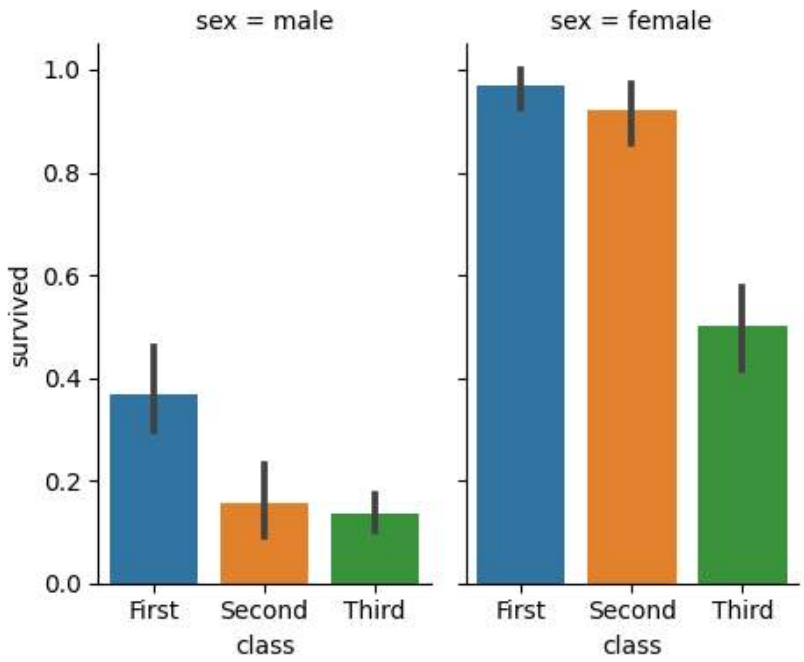
```
# Seaborn-based data visualization

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

print("Seaborn dataset : ", sns.get_dataset_names())

# load as Pandas data frame
df_titanic = sns.load_dataset("titanic")
print("\nSeaborn titanic :\n", df_titanic)

sns.barplot(x='class', y='age', data=df_titanic)
plt.show()
```



Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn catplot()

```
# Seaborn titanic, catplot(), col

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_titanic = sns.load_dataset("titanic")
#print("\nSeaborn titanic :\n", df_titanic)

sns.catplot(
    data=df_titanic, x="class", y="survived", col="sex",
    kind="bar", height=4, aspect=.6,
)
plt.show()
```

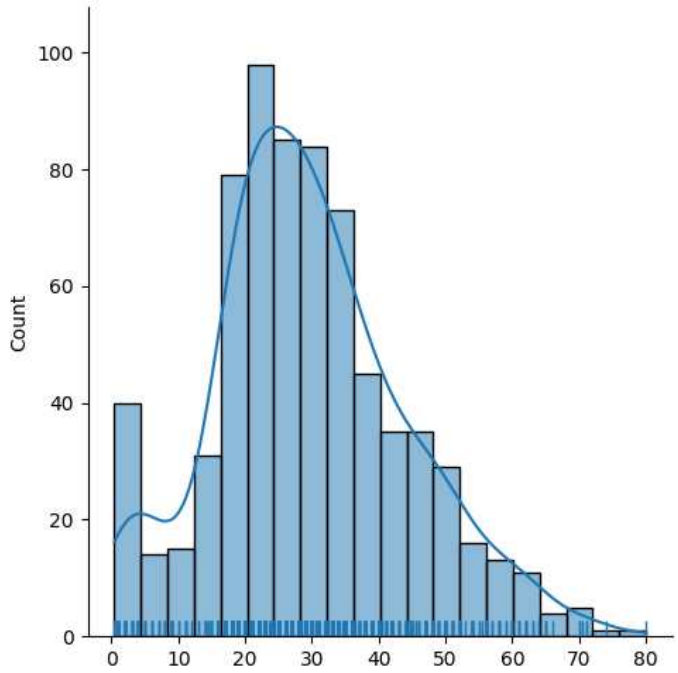


Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn displot()

- kde (kernel density estimation) : 분포 곡선 표시
- rug : x 축상에 작은 선분(rug)으로 데이터의 위치 표시

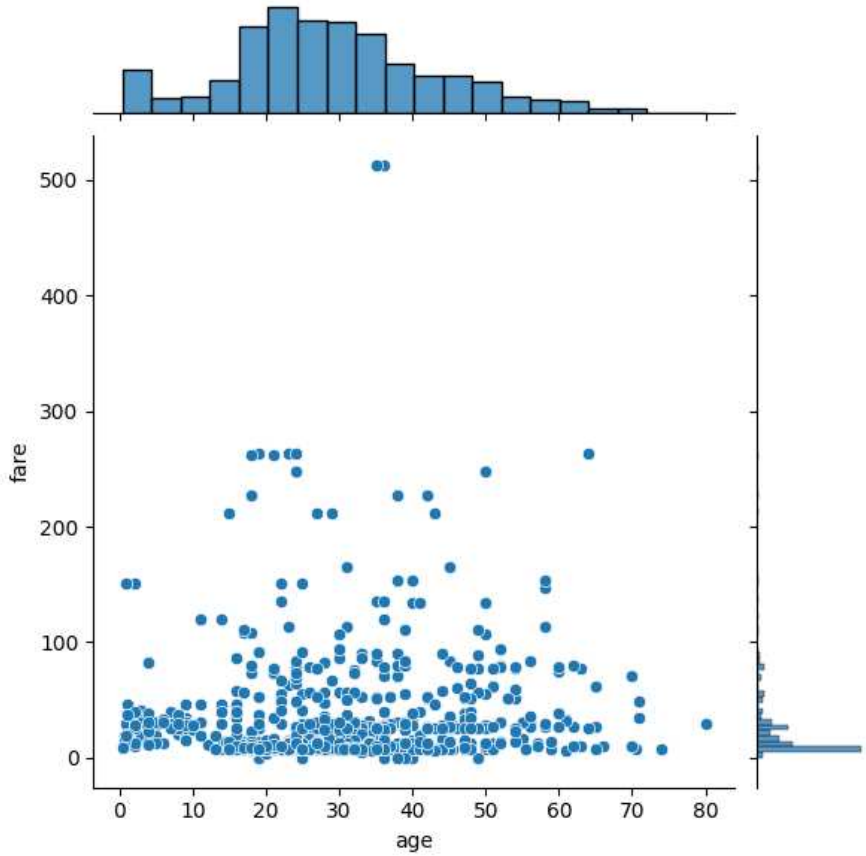
```
# Seaborn displot()

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_titanic = sns.load_dataset("titanic")
#print("\nSeaborn titanic :\n", df_titanic)

age = df_titanic.age.values
sns.displot(age, kde=True, rug=True)
plt.show()
```



Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn jointplot()

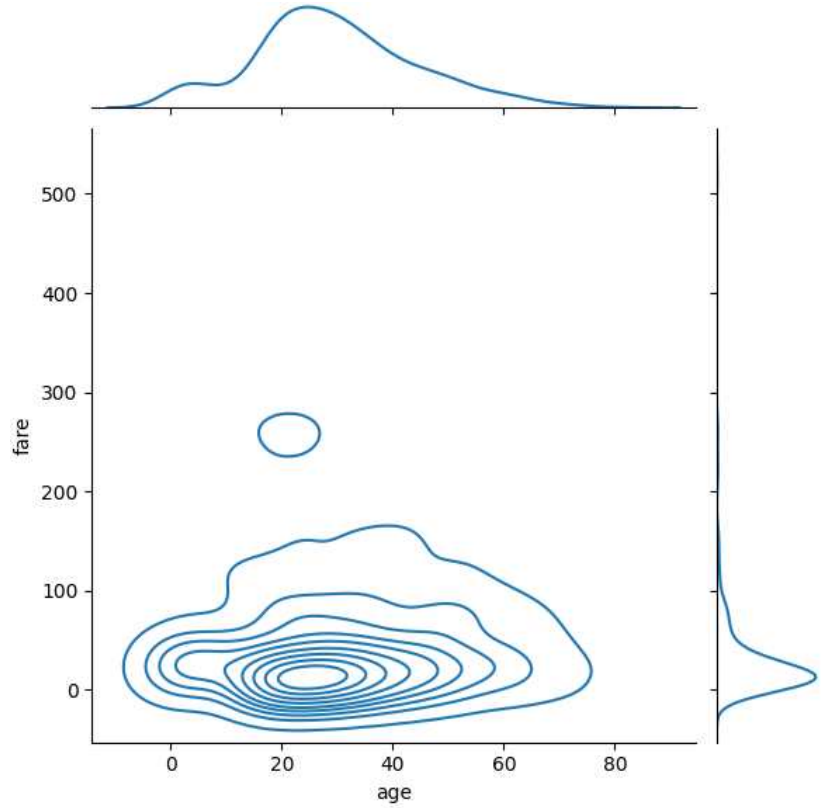
```
# Seaborn jointplot()

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_titanic = sns.load_dataset("titanic")
#print("\nSeaborn titanic :\n", df_titanic)

sns.jointplot(x="age", y="fare", data=df_titanic)
plt.show()
```



Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn jointplot(), kde

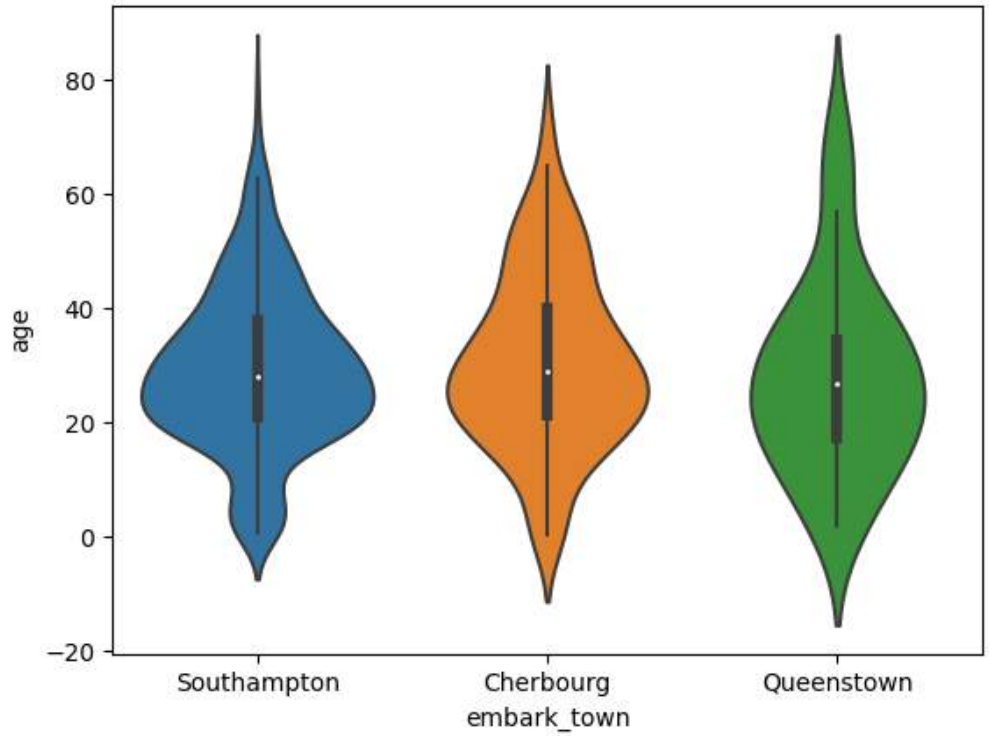
```
# Seaborn jointplot()

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_titanic = sns.load_dataset("titanic")
#print("\nSeaborn titanic :\n", df_titanic)

sns.jointplot(x="age", y="fare", kind="kde",\
 data=df_titanic)
plt.show()
```



Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn violinplot()

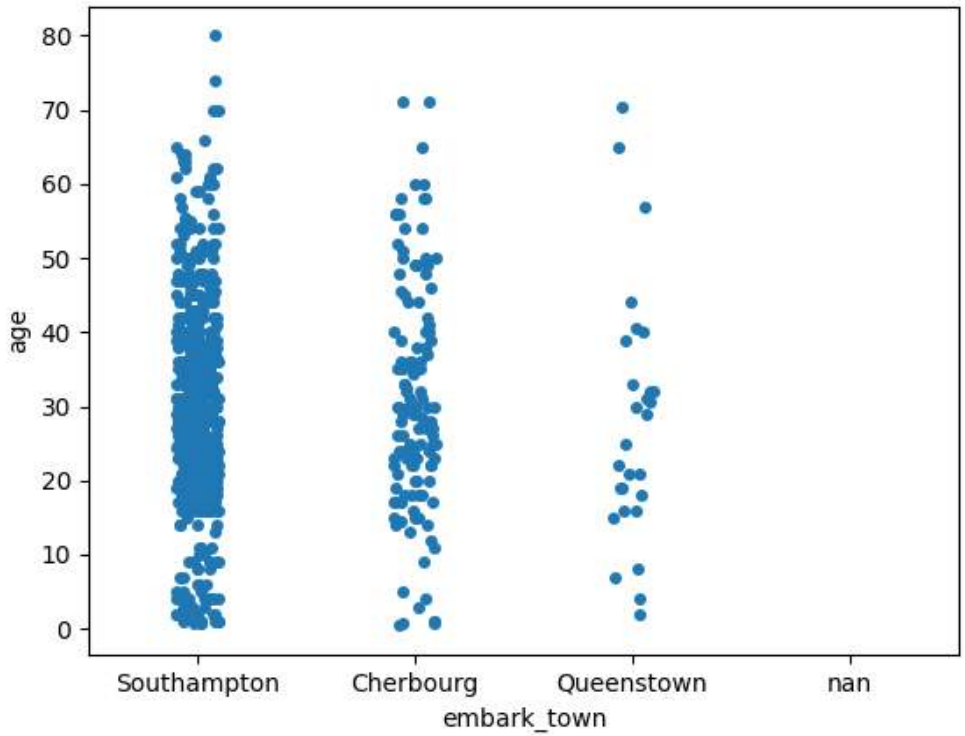
```
# Seaborn violinplot()

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_titanic = sns.load_dataset("titanic")
#print("\nSeaborn titanic :\n", df_titanic)

sns.violinplot(x="embark_town", y="age",\
 data=df_titanic)
plt.show()
```



Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn stripplot()

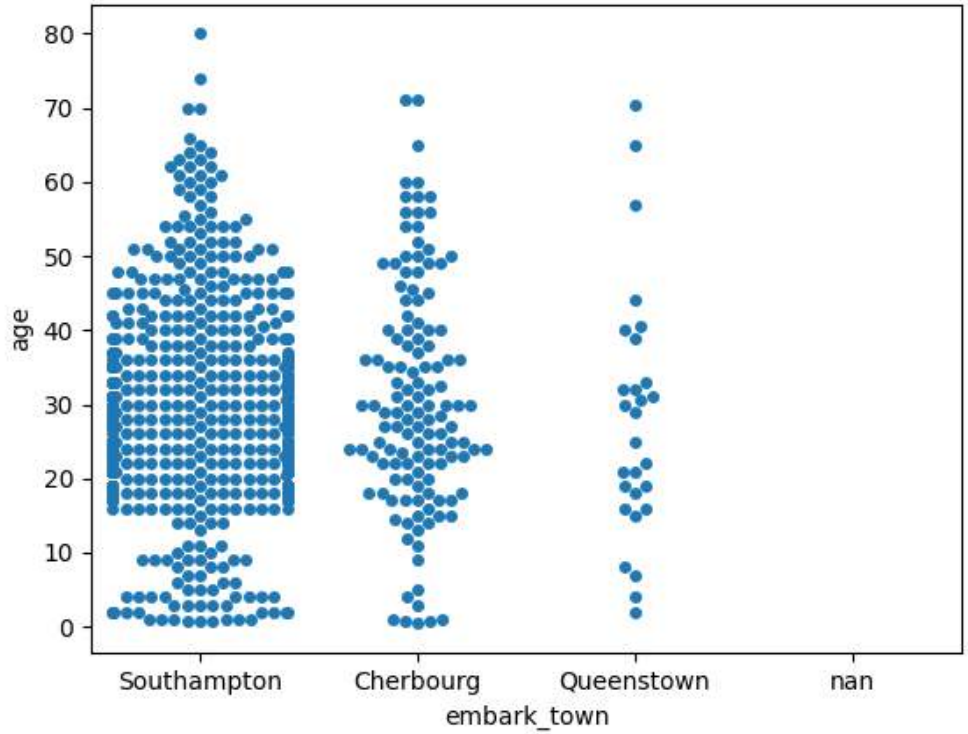
```
# Seaborn stripplot()

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_titanic = sns.load_dataset("titanic")
#print("\nSeaborn titanic :\n", df_titanic)

sns.stripplot(x="embark_town", y="age",\
 data=df_titanic)
plt.show()
```



Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn swarmpplot()

```
# Seaborn swarmplot()

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_titanic = sns.load_dataset("titanic")
#print("\nSeaborn titanic :\n", df_titanic)

sns.swarmplot(x="embark_town", y="age",\
 data=df_titanic)
plt.show()
```

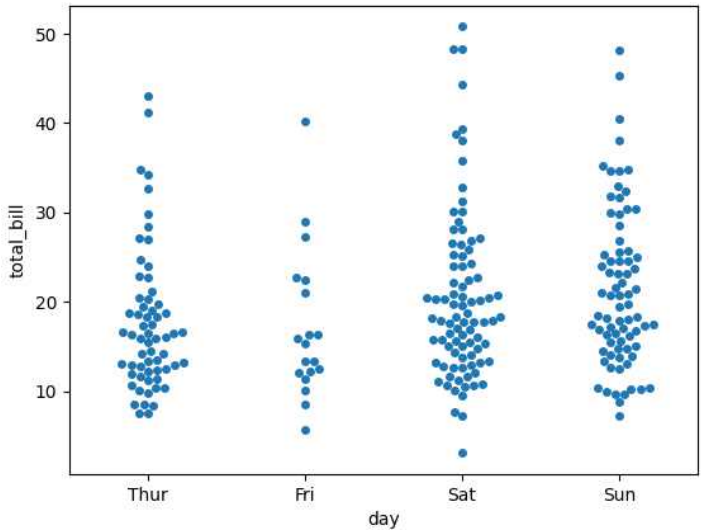


Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn swarmplot()

```
# Seaborn tips, swarmplot()

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_tips = sns.load_dataset("tips")
print("\nSeaborn tips :\n", df_tips)

sns.swarmplot(x="day", y="total_bill",\
 data=df_tips)
plt.show()
```

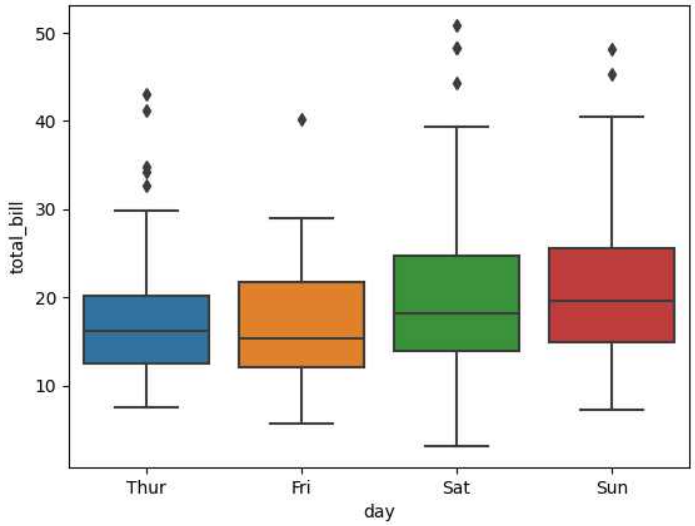
```
Seaborn tips :
     total_bill   tip     sex smoker   day    time  size
0         16.99  1.01  Female     No   Sun  Dinner     2
1         10.34  1.66    Male     No   Sun  Dinner     3
2         21.01  3.50    Male     No   Sun  Dinner     3
3         23.68  3.31    Male     No   Sun  Dinner     2
4         24.59  3.61  Female     No   Sun  Dinner     4
..          ...   ...     ...    ...   ...     ...   ...
239       29.03  5.92    Male     No   Sat  Dinner     3
240       27.18  2.00  Female    Yes   Sat  Dinner     2
241       22.67  2.00    Male    Yes   Sat  Dinner     2
242       17.82  1.75    Male     No   Sat  Dinner     2
243       18.78  3.00  Female     No  Thur  Dinner     2

[244 rows x 7 columns]
```



Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn boxplot()

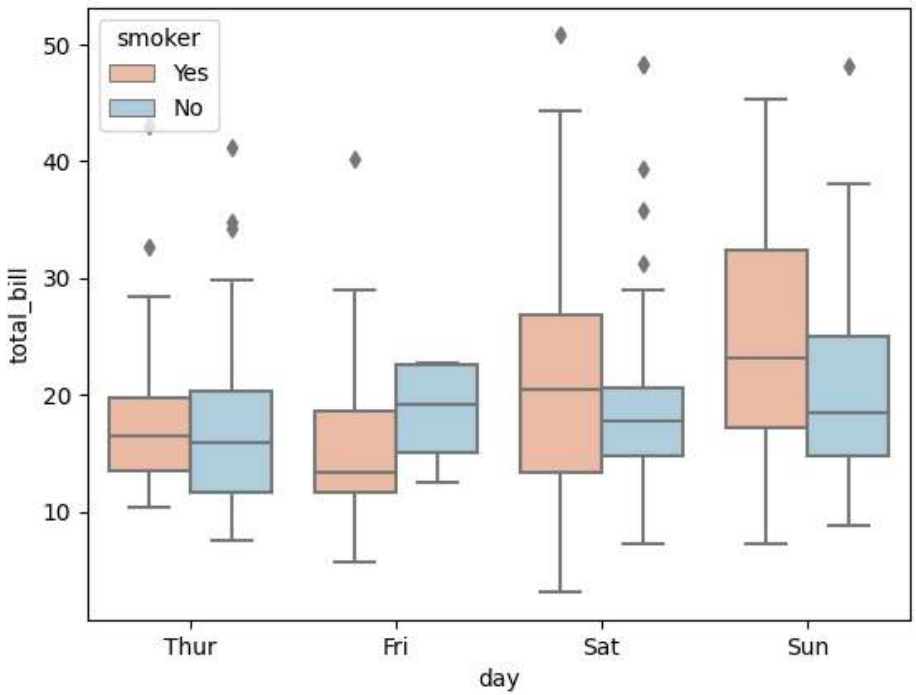
```
# Seaborn tips, boxplot()

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_tips = sns.load_dataset("tips")
#print("\nSeaborn tips :\n", df_tips)

sns.boxplot(x="day", y="total_bill", data=df_tips)
plt.show()
```



Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn boxplot(), hue, palette

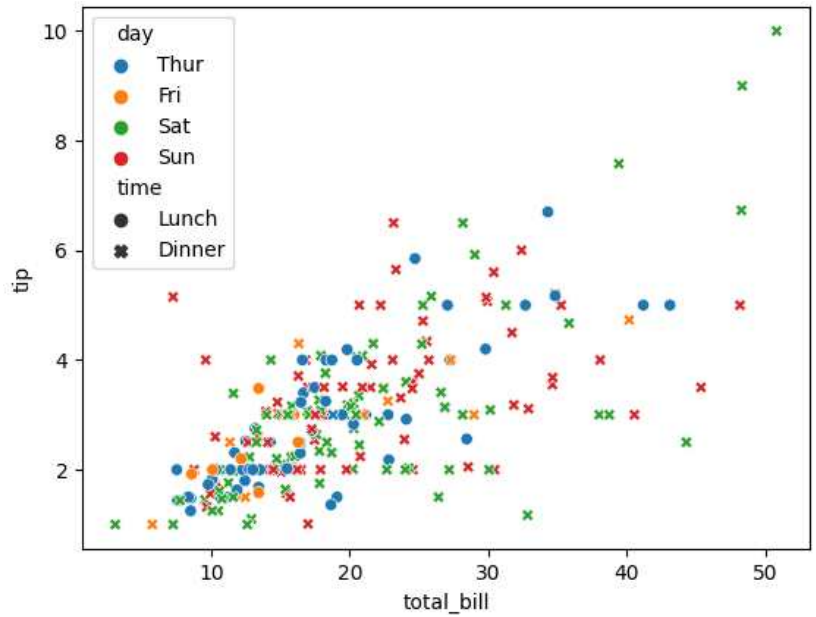
```
# Seaborn tips, boxplot(), hue

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_tips = sns.load_dataset("tips")
#print("\nSeaborn tips :\n", df_tips)

sns.boxplot(x="day", y="total_bill",\
 hue="smoker", palette = "RdBu", data=df_tips)
plt.show()
```



Seaborn 확장 모듈 기반 데이터 시각화

◆ Seaborn scatterplot(), hue, style

```
# Seaborn tips, scatterplot(), hue, style

import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load as Pandas data frame
df_tips = sns.load_dataset("tips")
#print("\nSeaborn tips :\n", df_tips)

sns.scatterplot(data=df_tips, x="total_bill",
 y="tip", hue="day", style="time")
plt.show()
```



Seaborn Pair Plot

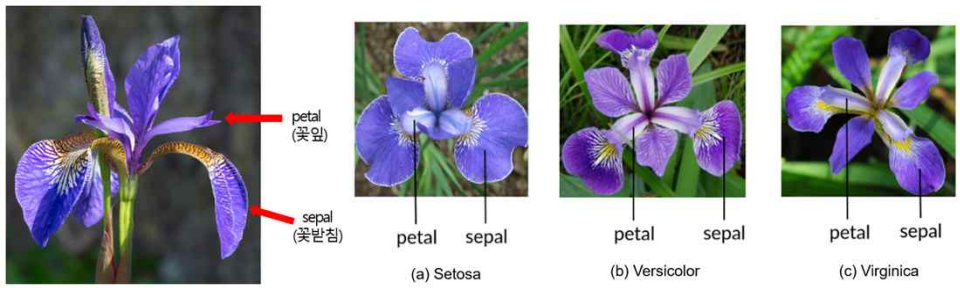
◆ Seaborn Pair Plot

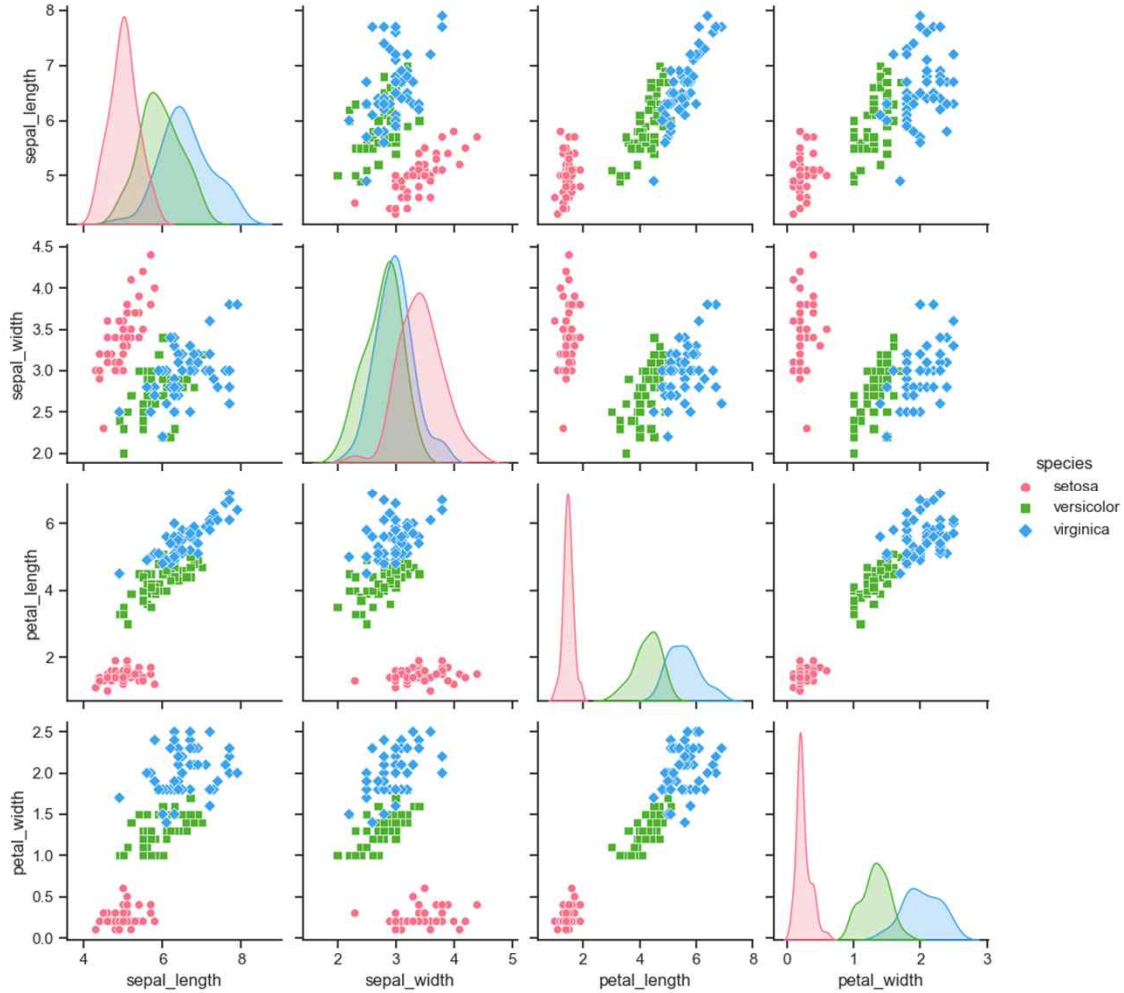
- sns.pairplot(data, hue = None, hue_order = None, palette = None, vars = None, x_vars = None, y_vars = None, kind = 'scatter', diag_kind = 'auto', markers = None, height = 2.5, aspect = 1, corner = False, dropna = False, plot_kws = None, diag_kws = None, grid_kws = None, size = None)

- Parameters:
  - data: It accepts the data depending on the visualization to be plotted.
  - hue: It is variable in data to map plot aspects to different colors.
  - hue_order: It is a list or string which is the order parameter of the hue variable in the palette.
  - palette: It is used to set the color of the hue.
  - dropna: It can drop missing values from the data before plotting.

Seaborn Pair Plot - iris (붓꽃)

```
# Seaborn pair plot for iris
import seaborn as sns
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style="ticks", color_codes=True)
df_iris = sns.load_dataset('iris')
#df_iris.drop('Id',axis=1,inplace=True)
print(df_iris.head())
#sns.pairplot(df_iris)
sns.pairplot(df_iris, hue="species",
    palette="husl", markers=["o", "s", "D"])
plt.show()
```





Homework 11

Homework 11.1

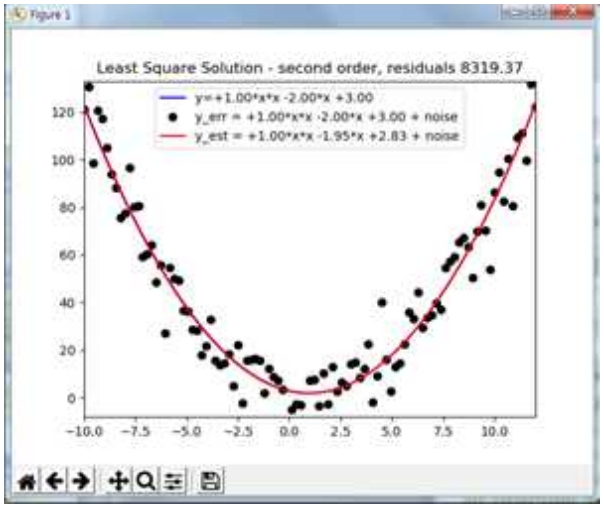
11.1 Matplotlib 기반 수평막대그래프 및 파이차트 (pie chart) 작성

- 2020년도 연령별 인구수를 파악하라. (참고자료: 연령별 인구현황 - https://jumin.mois.go.kr/ageStatMonth.do.)
- Matplotlib.pyplot의 barh() 메소드를 사용하여 수평막대그래프를 생성하고,
- pie() 메소드를 사용하여 파이차트를 생성하라.
- 참고자료: Matplotlib Tutorial - 파이썬으로 데이터 시각화하기, https://wikidocs.net/book/5011.

Homework 11.2

11.2 NumPy lstsq() - 최소 자승 오차 방정식 산출

- NumPy 패키지에 포함된 linspace() 함수를 사용하여 –10 ~+10 구간에서 100개의 등 간격 값을 가지의 원소들을 배열 X에 포함시켜라. 그리고 배열 X의 각 원소들에 대하여 $y(x) = x^2 - 2x + 3$ 의 공식에 따라 y 값을 계산하고, 이 y값에 random.normal(0, 10.0)으로 생성된 난수 값을 더하여, 배열 Y의 원소값으로 설정하라. 이렇게 준비된 배열 X와 배열 Y를 NumPy의 linalg 모듈의 lstsq() 함수를 사용하여 최소 자승 오차 방정식을 구하고, 그 계수들을 출력하라.
- (실행 예)



Homework 11.3

11.3 Pandas와 Excel 파일을 사용한 데이터 분석

- Excel 파일 (student_scores.xlsx)에 학생 10명의 국어, 영어, 수학, 과학 성적을 표로 준비하라.
  이 Excel 파일을 pandas의 read_excel() 함수를 사용하여 읽고, 데이터 프레임을 생성하라.
  각 학생들의 성적 평균을 계산하여 'Avg' 열을 추가하라.
- 데이터 프레임을 학생 성적 평균을 기준으로 내림 차순 정렬하라.
- 각 과목별 평균을 계산하여 'Total_Avg' 행을 추가하라.
- 각 학생별 성적 평균과 각 과목별 성적 평균을 추가한 데이터 프레임을 출력하라.
- 각 학생별 성적 평균과 각 과목별 성적 평균이 추가된 데이터 프레임을 Excel 파일 (processed_scores.xlsx)에 출력하라.
- (실행결과):

<Input Excel File>

	A	B	C	D	E	F
1	st_id	st_name	Eng	Kor	Math	Sci
2	1201	Kim	95.7	92.3	95.2	75.9
3	2202	Lee	92.4	94.5	93.5	92.4
4	1203	Park	85.7	88.7	90.3	87.3
5	1701	Yoon	76.8	80.2	83.5	75.4
6	1500	Choi	98.9	97.2	98.2	95.3
7	2512	Hong	80.2	95.7	75.9	92.3
8	3143	Hwang	94.2	92.4	92.4	94.5
9	4765	Song	87.3	85.7	87.3	88.7
10	5532	Kang	93.5	76.8	75.4	80.2
11	1276	Jung	96.3	98.9	95.3	97.2

<Output Excel File (Sorted, Total_Avg)>

	A	B	C	D	E	F	G	H
		st_id	st_name	Eng	Kor	Math	Sci	Avg
	4	1500	Choi	98.9	97.2	98.2	95.3	97.4
	9	1276	Jung	96.3	98.9	95.3	97.2	96.925
	6	3143	Hwang	94.2	92.4	92.4	94.5	93.375
	1	2202	Lee	92.4	94.5	93.5	92.4	93.2
	0	1201	Kim	95.7	92.3	95.2	75.9	89.775
	2	1203	Park	85.7	88.7	90.3	87.3	88
	7	4765	Song	87.3	85.7	87.3	88.7	87.25
	5	2512	Hong	80.2	95.7	75.9	92.3	86.025
	8	5532	Kang	93.5	76.8	75.4	80.2	81.475
	3	1701	Yoon	76.8	80.2	83.5	75.4	78.975
	10		Total_Avg	90.1	90.24	88.7	87.92	89.24

```
df =
    st_id st_name   Eng   Kor  Math   Sci
0    1201     Kim  95.7  92.3  95.2  75.9
1    2202     Lee  92.4  94.5  93.5  92.4
2    1203    Park  85.7  88.7  90.3  87.3
3    1701    Yoon  76.8  80.2  83.5  75.4
4    1500    Choi  98.9  97.2  98.2  95.3
5    2512    Hong  80.2  95.7  75.9  92.3
6    3143   Hwang  94.2  92.4  92.4  94.5
7    4765    Song  87.3  85.7  87.3  88.7
8    5532    Kang  93.5  76.8  75.4  80.2
9    1276    Jung  96.3  98.9  95.3  97.2

avgs_per_class =
Eng     90.10
Kor     90.24
Math    88.70
Sci     87.92
Avg     89.24
dtype: float64

df_sorted_with_avg =
     st_id    st_name   Eng    Kor  Math    Sci     Avg
4   1500.0       Choi  98.9  97.20  98.2  95.30  97.400
9   1276.0       Jung  96.3  98.90  95.3  97.20  96.925
6   3143.0      Hwang  94.2  92.40  92.4  94.50  93.375
1   2202.0        Lee  92.4  94.50  93.5  92.40  93.200
0   1201.0        Kim  95.7  92.30  95.2  75.90  89.775
2   1203.0       Park  85.7  88.70  90.3  87.30  88.000
7   4765.0       Song  87.3  85.70  87.3  88.70  87.250
5   2512.0       Hong  80.2  95.70  75.9  92.30  86.025
8   5532.0       Kang  93.5  76.80  75.4  80.20  81.475
3   1701.0       Yoon  76.8  80.20  83.5  75.40  78.975
10     NaN  Total_Avg  90.1  90.24  88.7  87.92  89.240
Writing df to excel file
```

Homework 11.4 Seaborn을 사용한 시각화

11.4 Seaborn을 사용한 데이터 시각화

- Seaborn의 dataset iris를 Pandas data frame으로 설정하고, 이 data frame의 Seaborn pair plot를 생성하라.