Python Packages

```
import warnings
warnings.filterwarnings('ignore')
```

I. numpy

- 수학 및 과학적 연산을 쉽고 빠르게 지원
- 다차원 행열(Array)을 효과적으로 처리

1) numpy Package import

```
import numpy as np
```

2) Array 생성

(1) Scalar - 0D Array - Rank0 Tensor

```
a0 = np.array(9)
```

```
print(a0)
```

```
9
```

(2) Vector - 1D Array - Rank1 Tensor

```
a1 = np.array([1, 3, 5, 7, 9])
```

```
print(a1)
```

```
[1 3 5 7 9]
```

(3) Matrix - 2D Array - Rank2 Tensor

```
a2 = np.array([[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]])
```

```
print(a2)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

(4) Array - 3D Array - Rank3 Tensor

```
a3 = np.array([[[1, 2],
                [3, 4]],
               [[5, 6],
                [7, 8]],
               [[9, 10],
                [11, 12]]])
```

```
print(a3)
```

```
[[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
```

3) AR.shape and AR.reshape( )

```
AR = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
```

```
print(AR)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12]
```

(1) .shape

```
AR.shape
```

```
(12,)
```

(2) .reshape(3, 4)

- .reshape(행, 열)

```
AR2 = AR.reshape(3, 4)
```

```
AR2 = AR.reshape(3, 4)
```

```
print(AR2)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

```
AR2.shape
```

```
(3, 4)
```

> (3) .reshape(3, 2, 2)

- .reshape(축, 행, 열)

```
AR3 = AR.reshape(3, 2, 2)
```

```
print(AR3)
```

```
[[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
```

```
AR3.shape
```

```
(3, 2, 2)
```

4) 범위 지정(arange) 함수 사용

> (1) 10개의 연속된 값 생성

```
np.arange(10)
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

> (2) 1부터 9까지 1간격으로 생성

```
np.arange(1, 10)
```

```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

(3) 1부터 9까지 2간격으로 생성

```
np.arange(1, 10, 2)
```

```
array([1, 3, 5, 7, 9])
```

(4) 생성된 Array에 .reshape( ) 적용

```
np.arange(1, 10).reshape(3, 3)
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

5) 특별한 형태의 Array 생성

(1) 0과 1로만 구성된 Array

```
np.zeros(9)
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

```
np.ones(9)
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

```
np.zeros([3, 4])
```

```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

```
np.ones([4, 3])
```

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

(2) 3 x 3 단위행렬

```
np.eye(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
```

```
       [0., 0., 1.]])
```

(3) 난수 Array 생성

```
np.random.rand(3, 2, 2)
```

```
array([[[0.90263998, 0.6033992 ],
        [0.03030465, 0.02533559]],

       [[0.68642969, 0.68016323],
        [0.12076552, 0.4761034 ]],

       [[0.88153973, 0.25818098],
        [0.72321264, 0.24322276]]])
```

```
np.random.randint(1, 45, size = (5, 6))
```

```
array([[20, 13, 27, 13, 25, 34],
       [12, 17, 22,  9,  9, 28],
       [37,  7, 32, 23, 40,  6],
       [37, 22, 40, 19, 28, 33],
       [ 5, 18,  3, 27, 35, 41]])
```

```
np.random.seed(2045)

np.random.choice(np.arange(1, 46), size = (5 ,6), replace = False)
```

```
array([[ 7, 32,  6, 41,  4, 34],
       [30, 28, 16,  5, 38, 33],
       [31, 13, 25, 23, 43, 12],
       [45,  8, 29, 22, 18,  9],
       [21, 20, 40,  2, 37, 39]])
```

6) Array 연산

```
a1 = np.array([1, 3, 5, 7, 9])
a2 = np.array([10, 30, 50, 70, 90])
```

(1) 기본 연산

```
a1 + a2
```

```
array([11, 33, 55, 77, 99])
```

```
a2 - a1
```

```
array([ 9, 27, 45, 63, 81])
```

```
a1 * a2
```

```
array([ 10,  90, 250, 490, 810])
```

```
a2 / a1
```

```
array([10., 10., 10., 10., 10.])
```

```
a1 * 3
```

```
array([ 3,  9, 15, 21, 27])
```

```
a1 ** 2
```

```
array([ 1,  9, 25, 49, 81])
```

(2) 통계량 연산

- 총합

```
a1.sum()
```

```
25
```

- 평균

```
a2.mean()
```

```
50.0
```

- 분산

```
a2.var()
```

```
800.0
```

- 표준 편차

```
a2.std()
```

```
28.284271247461902
```

- 최소값

```
a2.min()
```

```
10
```

- 최대값

```
a2.max()
```

```
90
```

- 누적(Cumulative)합

```
a1.cumsum()
```

```
array([ 1,  4,  9, 16, 25])
```

- 누적(Cumulative)곱

```
a1.cumprod()
```

```
array([  1,   3,  15, 105, 945])
```

7) Matrix 연산

- A1, A2 지정

```
A1 = np.array([2, 4, 6, 8]).reshape(2, 2)
```

```
print(A1)
```

```
[[2 4]
 [6 8]]
```

```
A2 = np.array([3, 5, 7, 9]).reshape(2, 2)
```

```
print(A2)
```

```
[[3 5]
 [7 9]]
```

(1) Matrix 곱 -> A1 @ A2

- A1 @ A2

```
A1.dot(A2)
```

```
array([[ 34,  46],
       [ 74, 102]])
```

- A2 @ A1

```
np.dot(A2, A1)
```

```
array([[ 36,  52],
       [ 68, 100]])
```

- Warning : A1 * A2

```
A1 * A2
```

```
array([[ 6, 20],
       [42, 72]])
```

> (2) 전치 행렬

- A1의 전치 행렬

```
np.transpose(A1)
```

```
array([[2, 6],
       [4, 8]])
```

- A2의 전치 행렬

```
A2.transpose()
```

```
array([[3, 7],
       [5, 9]])
```

II. pandas

1) File Upload to Colab

- Colab 가상환경에 파일 올리기
- Colab 종료 시 파일은 삭제됨
- Local_Disk to Colab_Linux_File_System

- PII.csv & PII.xlsx

- 업로드된 파일 확인

```
!ls -l
```

```
total 20
-rw-r--r-- 1 root root   723 Feb  6 23:03 PII.csv
-rw-r--r-- 1 root root 11370 Feb  6 23:03 PII.xlsx
drwxr-xr-x 1 root root  4096 Feb  4 15:26 sample_data
```

2) pandas - DataFrame

(1) with PII.csv

```
import pandas as pd
```

```
DF1 = pd.read_csv('PII.csv')
```

```
DF1
```

	Name	Gender	Age	Grade	Picture	BloodType	Height	Weight
0	송태섭	남자	21	3	무	B	179.1	63.9
1	최유정	여자	23	1	유	A	177.1	54.9
2	이한나	여자	20	1	무	A	167.9	50.2
3	김소혜	여자	23	3	무	O	176.1	53.5
4	서태웅	남자	24	4	무	B	176.1	79.8
5	정대만	남자	24	2	유	B	175.2	61.7
6	이정환	남자	22	4	무	B	169.1	69.8
7	채소연	여자	22	2	유	AB	169.9	52.7
8	강백호	남자	23	3	무	O	165.5	68.5
9	전소미	여자	22	2	유	O	161.9	52.3
10	변덕규	남자	21	1	무	A	163.2	55.5
11	정채연	여자	22	2	무	B	157.8	44.9
12	권준호	남자	24	4	유	O	166.9	61.7
13	채치수	남자	23	3	무	AB	181.8	85.9
14	윤대협	남자	22	2	유	AB	180.3	76.2
15	김세정	여자	21	1	무	O	155.5	44.9
16	서중석	남자	23	1	무	A	168.9	62.7

(2) with PII.xlsx

```
DF2 = pd.read_excel('PII.xlsx')
```

```
DF2
```

	Name	Gender	Age	Grade	Picture	BloodType	Height	Weight
0	송태섭	남자	21	3	무	B	179.1	63.9
1	최유정	여자	23	1	유	A	177.1	54.9
2	이한나	여자	20	1	무	A	167.9	50.2
3	김소혜	여자	23	3	무	O	176.1	53.5
4	서태웅	남자	24	4	무	B	176.1	79.8
5	정대만	남자	24	2	유	B	175.2	61.7
6	이정환	남자	22	4	무	B	169.1	69.8
7	채소연	여자	22	2	유	AB	169.9	52.7
8	강백호	남자	23	3	무	O	165.5	68.5
9	전소미	여자	22	2	유	O	161.9	52.3
10	변덕규	남자	21	1	무	A	163.2	55.5
11	정채연	여자	22	2	무	B	157.8	44.9
12	권준호	남자	24	4	유	O	166.9	61.7
13	채치수	남자	23	3	무	AB	181.8	85.9
14	윤대협	남자	22	2	유	AB	180.3	76.2
15	김세정	여자	21	1	무	O	155.5	44.9
16	서중석	남자	22	1	무	A	168.9	62.7

(3) Information

```
type(DF1)
```

```
pandas.core.frame.DataFrame
```

```
DF1.index
```

```
RangeIndex(start=0, stop=17, step=1)
```

```
DF1.columns
```

```
Index(['Name', 'Gender', 'Age', 'Grade', 'Picture', 'BloodType', 'Height',
```

```
       'Weight'],
      dtype='object')
```

```
DF1.values
```

```
array([['송태섭', '남자', 21, 3, '무', 'B', 179.1, 63.9],
       ['최유정', '여자', 23, 1, '유', 'A', 177.1, 54.9],
       ['이한나', '여자', 20, 1, '무', 'A', 167.9, 50.2],
       ['김소혜', '여자', 23, 3, '무', 'O', 176.1, 53.5],
       ['서태웅', '남자', 24, 4, '무', 'B', 176.1, 79.8],
       ['정대만', '남자', 24, 2, '유', 'B', 175.2, 61.7],
       ['이정환', '남자', 22, 4, '무', 'B', 169.1, 69.8],
       ['채소연', '여자', 22, 2, '유', 'AB', 169.9, 52.7],
       ['강백호', '남자', 23, 3, '무', 'O', 165.5, 68.5],
       ['전소미', '여자', 22, 2, '유', 'O', 161.9, 52.3],
       ['변덕규', '남자', 21, 1, '무', 'A', 163.2, 55.5],
       ['정채연', '여자', 22, 2, '무', 'B', 157.8, 44.9],
       ['권준호', '남자', 24, 4, '유', 'O', 166.9, 61.7],
       ['채치수', '남자', 23, 3, '무', 'AB', 181.8, 85.9],
       ['윤대협', '남자', 22, 2, '유', 'AB', 180.3, 76.2],
       ['김세정', '여자', 21, 1, '무', 'O', 155.5, 44.9],
       ['신준섭', '남자', 23, 1, '무', 'A', 168.9, 62.7]], dtype=object)
```

```
DF1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17 entries, 0 to 16
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Name       17 non-null     object 
 1   Gender     17 non-null     object 
 2   Age        17 non-null     int64  
 3   Grade      17 non-null     int64  
 4   Picture    17 non-null     object 
 5   BloodType  17 non-null     object 
 6   Height     17 non-null     float64
 7   Weight     17 non-null     float64
dtypes: float64(2), int64(2), object(4)
memory usage: 1.2+ KB
```

(4) Function

```
DF1.head()
```

	Name	Gender	Age	Grade	Picture	BloodType	Height	Weight
0	송태섭	남자	21	3	무	B	179.1	63.9
1	최유정	여자	23	1	유	A	177.1	54.9
2	이한나	여자	20	1	무	A	167.9	50.2
3	김소혜	여자	23	3	무	O	176.1	53.5
4	서태웅	남자	24	4	무	B	176.1	79.8

```
DF1[0:5]
```

	Name	Gender	Age	Grade	Picture	BloodType	Height	Weight
0	송태섭	남자	21	3	무	B	179.1	63.9
1	최유정	여자	23	1	유	A	177.1	54.9
2	이한나	여자	20	1	무	A	167.9	50.2
3	김소혜	여자	23	3	무	O	176.1	53.5
4	서태웅	남자	24	4	무	B	176.1	79.8

```
DF1.tail(3)
```

	Name	Gender	Age	Grade	Picture	BloodType	Height	Weight
14	윤대협	남자	22	2	유	AB	180.3	76.2
15	김세정	여자	21	1	무	O	155.5	44.9
16	신준섭	남자	23	1	무	A	168.9	62.7

```
DF1[-3:]
```

	Name	Gender	Age	Grade	Picture	BloodType	Height	Weight
14	윤대협	남자	22	2	유	AB	180.3	76.2
15	김세정	여자	21	1	무	O	155.5	44.9
16	신준섭	남자	23	1	무	A	168.9	62.7

```
DF1.describe()
```

	Age	Grade	Height	Weight
count	17.000000	17.000000	17.000000	17.000000
mean	22.352941	2.294118	170.135294	61.123529
std	1.169464	1.104802	7.853896	11.867894
min	20.000000	1.000000	155.500000	44.900000
25%	22.000000	1.000000	165.500000	52.700000
50%	22.000000	2.000000	169.100000	61.700000
75%	23.000000	3.000000	176.100000	68.500000
max	24.000000	4.000000	181.800000	85.900000

```
DF1.mean()
```

```
Age        22.352941
```

```
Grade       2.294118
Height    170.135294
Weight     61.123529
dtype: float64
```

3) pandas - Series

(1) Information

```
type(DF1['Height'])
```

```
pandas.core.series.Series
```

```
DF1[['Height', 'Age']]
```

	Height	Age
0	179.1	21
1	177.1	23
2	167.9	20
3	176.1	23
4	176.1	24
5	175.2	24
6	169.1	22
7	169.9	22
8	165.5	23
9	161.9	22
10	163.2	21
11	157.8	22
12	166.9	24
13	181.8	23
14	180.3	22
15	155.5	21
16	168.9	22

```
DF1.Height
```

```
0    179.1
1    177.1
2    167.9
3    176.1
4    176.1
5    175.2
```

```
6     169.1
7     169.9
8     165.5
9     161.9
10    163.2
11    157.8
12    166.9
13    181.8
14    180.3
15    155.5
16    168.9
Name: Height, dtype: float64
```

(2) Function

```
DF1['Height'].sum()
```

```
2892.2999999999997
```

```
DF1['Height'].mean()
```

```
170.1352941176471
```

```
DF1['Height'].var()
```

```
61.683676470588225
```

```
DF1['Height'].std()
```

```
7.8538956238664275
```

```
DF1.Height.min()
```

```
155.5
```

```
DF1.Height.max()
```

```
181.8
```

```
DF1.groupby(['BloodType']).mean()
```

BloodType	Age	Grade	Height	Weight
A	21.750000	1.000000	169.275000	55.825
AB	22.333333	2.333333	177.333333	71.600
B	22.600000	3.000000	171.460000	64.020
O	22.600000	2.600000	165.180000	56.180

```
DF1.groupby(['BloodType'])['Height'].mean()
```

```
BloodType
A     169.275000
AB    177.333333
B     171.460000
O     165.180000
Name: Height, dtype: float64
```

(3) Indexing & Slicing

```
DF1.Height[16]
```

```
168.9
```

```
DF1.Height[0:5]
```

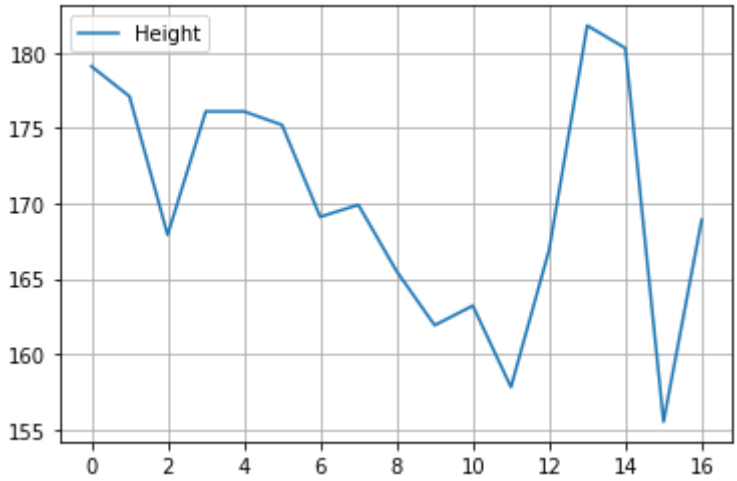
```
0    179.1
1    177.1
2    167.9
3    176.1
4    176.1
Name: Height, dtype: float64
```

4) pandas - Visualization

(1) 선 그래프

```
DF1[['Height']].plot(kind = 'line', grid = True)
```

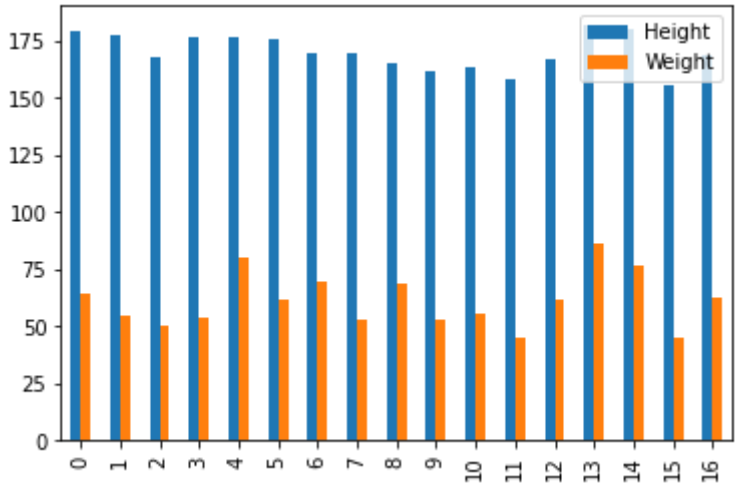
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd7e321f320>
```



(2) 막대 그래프

```
DF1[['Height', 'Weight']].plot(kind = 'bar')
```

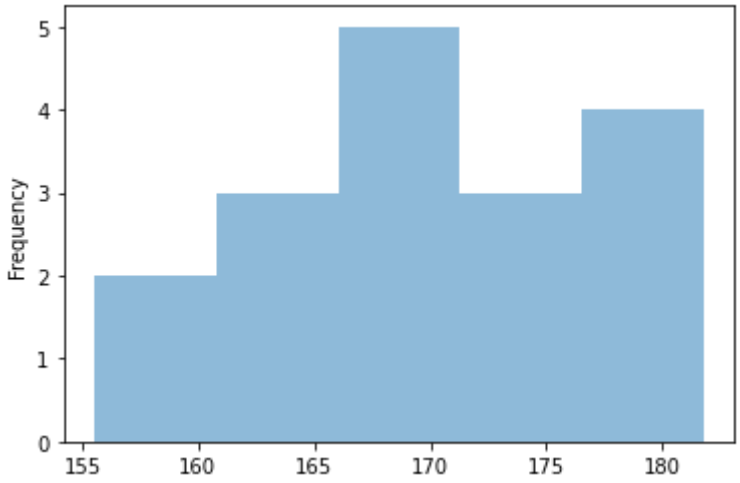
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd7e317c7f0>
```



(3) 히스토그램

```
DF1['Height'].plot(kind = 'hist', bins = 5, alpha = 0.5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd7e2cebb38>
```



(4) 상자 그래프

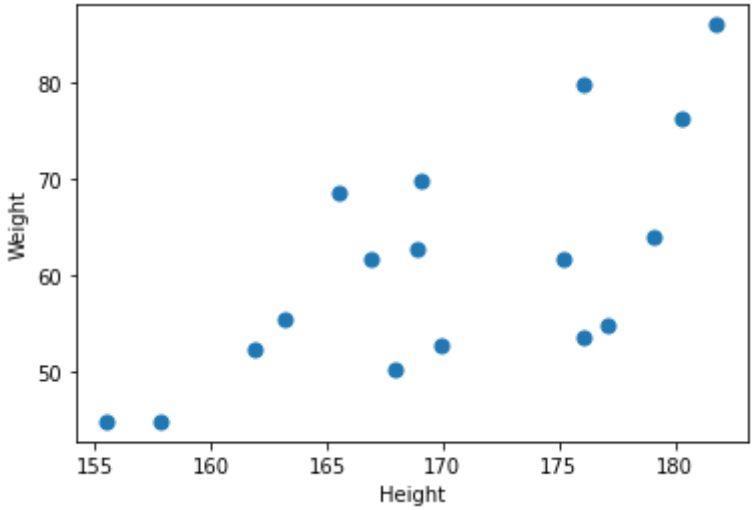
```
DF1['Height'].plot(kind = 'box')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd7e2bf1160>
```

(5) 산점도

```
DF1[['Height', 'Weight']].plot(kind = 'scatter',
                               x = 'Height', y = 'Weight',
                               s = 50)
```

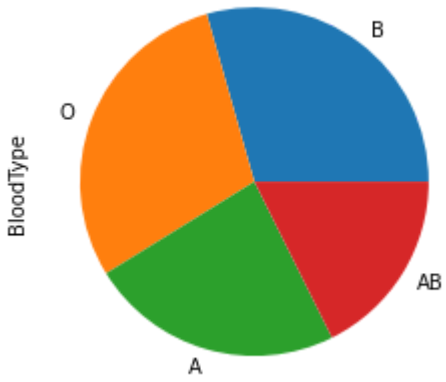
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd7e2af7dd8>
```



(6) 파이 그래프

```
DF1.BloodType.value_counts().plot(kind = 'pie')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd7e2adcba8>
```



III. matplotlib

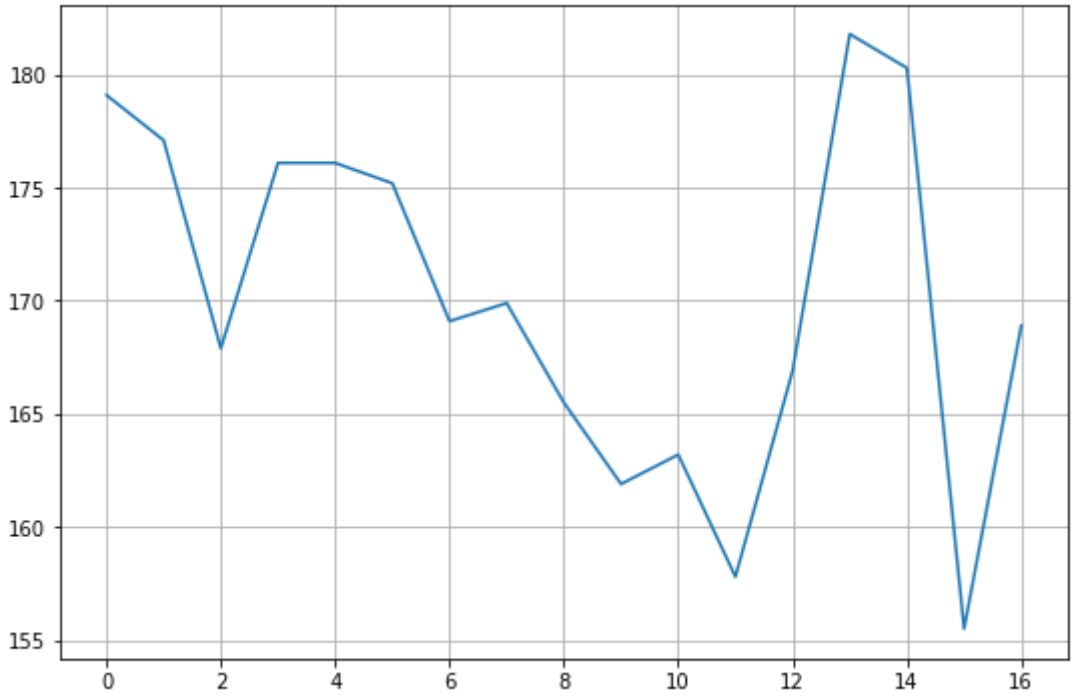
```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
DF1 = pd.read_csv('PII.csv')
```

1) 선 그래프

```
plt.figure(figsize = (9, 6))
plt.plot(DF1.Height)
plt.grid(True)
plt.show()
```

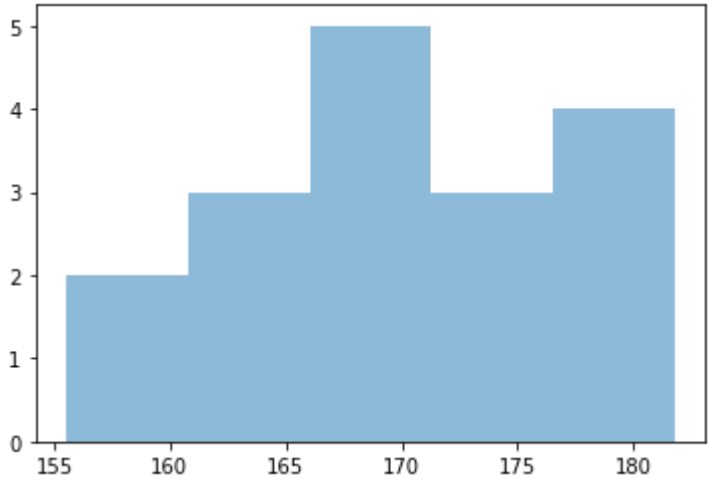


2) 막대 그래프

```
plt.bar(DF1.index, DF1.Height, width = 0.3, color = 'g')
plt.show()
```

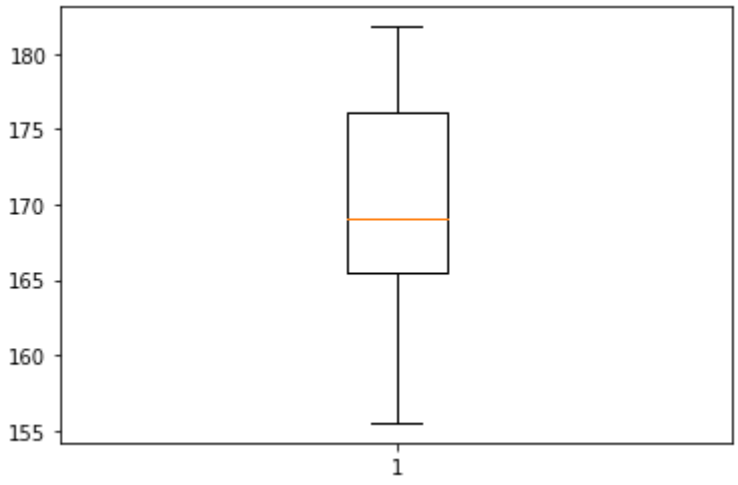
3) 히스토그램

```
plt.hist(DF1.Height, bins = 5, alpha = 0.5)
plt.show()
```



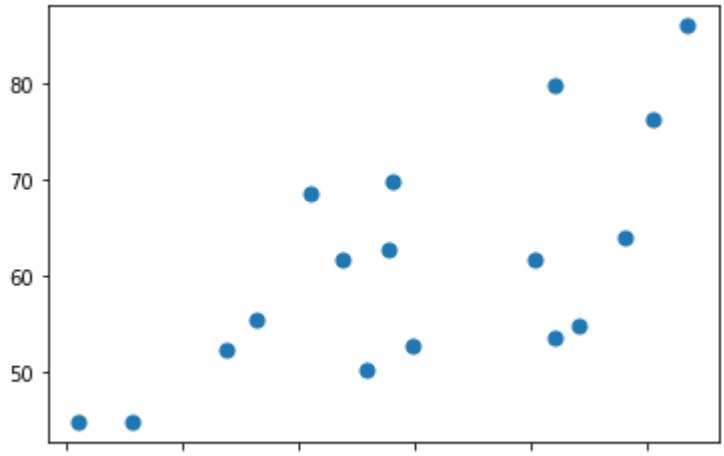
4) 상자 그래프

```
plt.boxplot(DF1.Height)
plt.show()
```



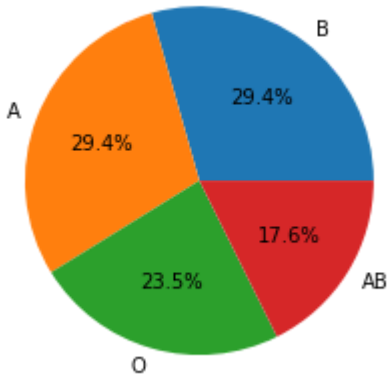
5) 산점도

```
plt.scatter(DF1.Height, DF1.Weight, s = 50)
plt.show()
```



6) 파이 그래프

```
plt.pie(DF1.BloodType.value_counts(),
        labels = DF1.BloodType.unique(),
        autopct = '%.1f%%')
plt.show()
```

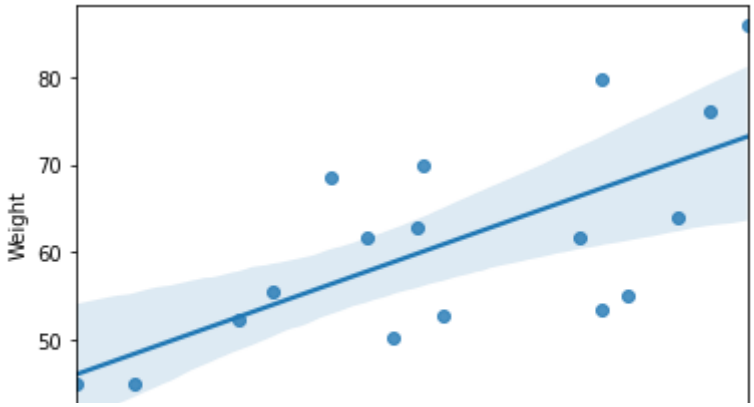


IV. seaborn

1) 회귀선 그리기

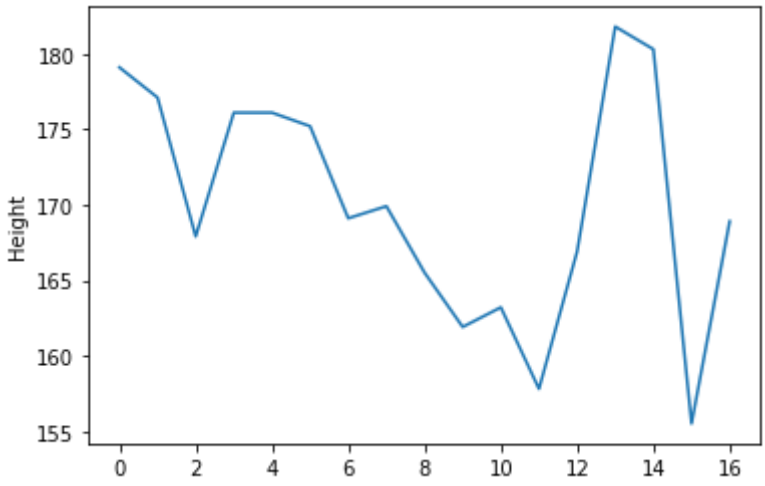
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
sns.regplot(DF1.Height, DF1.Weight)
plt.show()
```



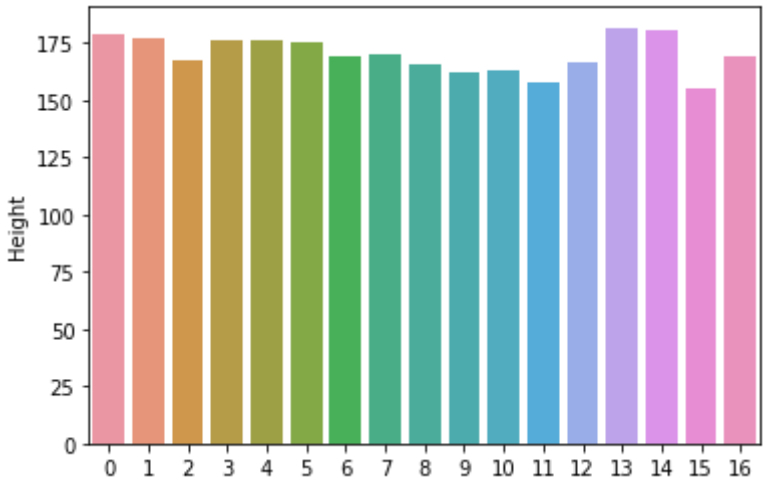
2) 선 그래프

```
sns.lineplot(DF1.index, DF1.Height)
plt.show()
```



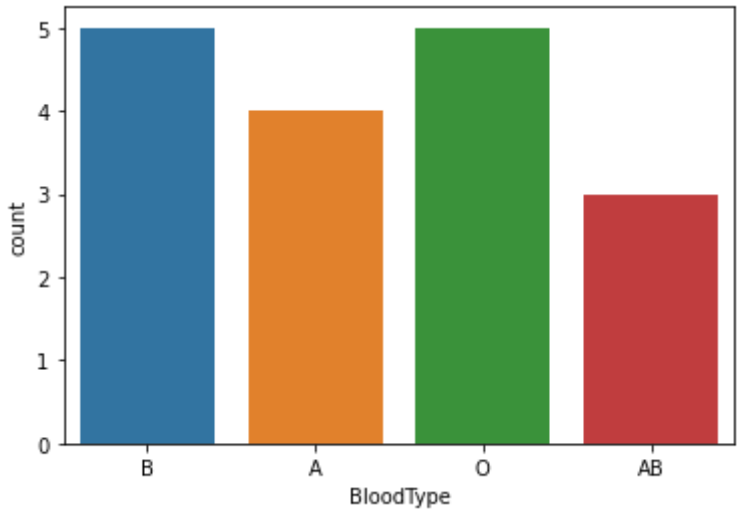
3) 막대 그래프-연속형

```
sns.barplot(x = DF1.index, y = DF1.Height)
plt.show()
```



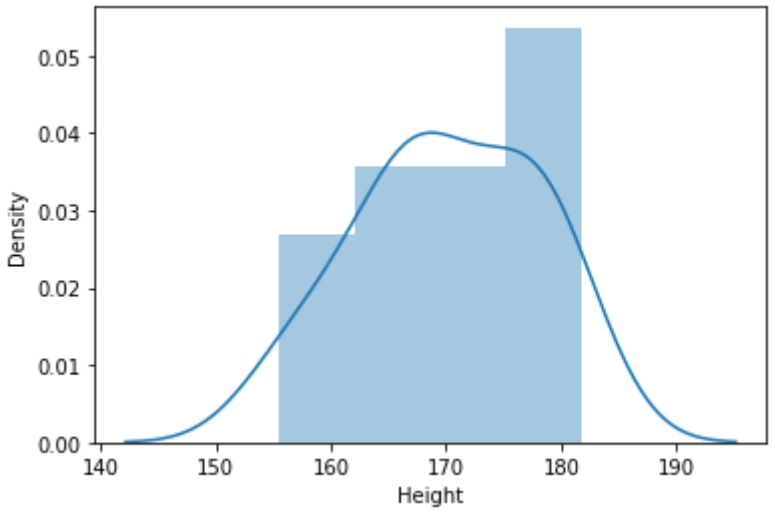
4) 막대 그래프-명목형

```
sns.countplot(DF1.BloodType)
plt.show()
```



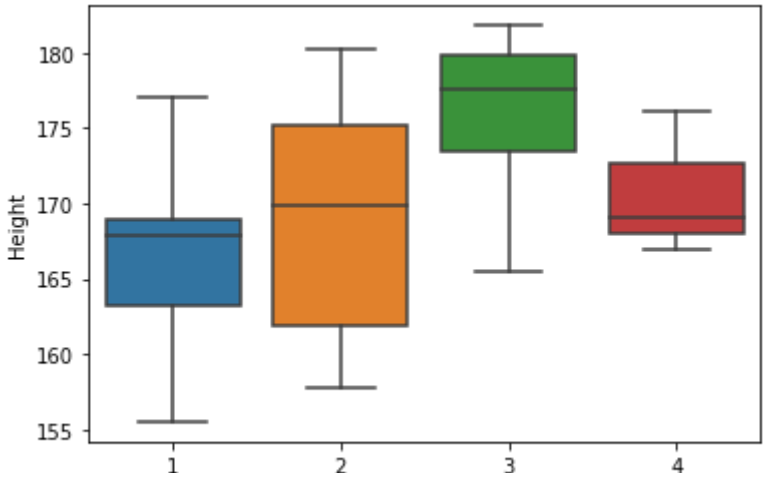
5) 히스토그램

```
sns.distplot(DF1.Height)
plt.show()
```



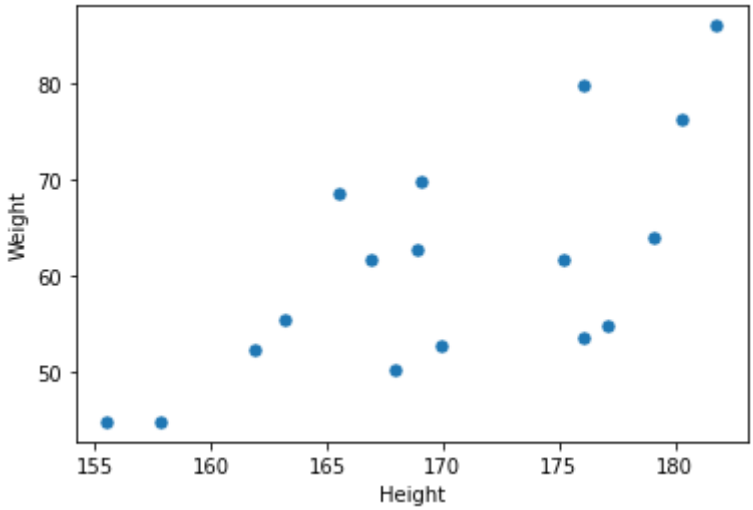
6) 상자 그래프

```
sns.boxplot(y = DF1.Height, x = DF1.Grade)
plt.show()
```



7) 산점도

```
sns.scatterplot(DF1.Height, DF1.Weight, s = 50)
plt.show()
```



#

#

#

The End

#

#

#