Practice #5

- The midterm examination score of 50 students are shown as the following table

Score	Frequency
1 – 5	2
6 – 10	6
11 – 15	13
16 – 20	17
21 – 25	12

Find mean, variance, skewness, kurtosis, 1st quartile, 3rd quartile and interquartile range of midterm score.

Solution (mean and variance)

```
data = [3, 8, 13, 18, 23]
weights = [2, 6, 13, 17, 12]
import numpy as np

def weighted_mean(data, weights):
    return np.average(data, weights = weights)

print(weighted_mean(data, weights))

def weighted_var(data, weights):
    return np.average((data - weighted_mean(data, weights))**2, weights = weights)

print(weighted_var(data, weights))
```

Solution (Skewness and Kurtosis)

```
def weighted_skew(data, weights):
    return (np.average((data - weighted_mean(data, weights))**3, weights=weights)/
            weighted_var(data, weights)**1.5)

print(weighted_skew(data, weights))

def weighted_kurt(data, weights):
    return (np.average((data - weighted_mean(data, weights))**4, weights = weights)/
            weighted_var(data, weights)**2)

print(weighted_kurt(data, weights))
```

Solution (Quantile and IQR)

```
def weighted_percentile(data, percents, weights=None):
    ind=np.argsort(data)
    d=data[ind]
    w=weights[ind]
    p=1.*w.cumsum()/w.sum()*100
    y=np.interp(percents, p, d)
    return y
first_quantile = weighted_percentile(np.array([3,8,13,18,23]),25,weights=np.array([2,6,13,17,12]))
third_quantile = weighted_percentile(np.array([3,8,13,18,23]),75,weights=np.array([2,6,13,17,12]))
IQR = third_quantile - first_quantile
print(first_quantile, third_quantile, IQR)
```

Practice #6

- Assume the heights (y) and weights (x) of a certain data

Heights	156	167	173	178	189	194	171	185
Weights	53	59	65	78	82	79	84	77

- Plot the scatter plot of Heights and Weights
- Find the regression model
- Find Mean Square Error (MSE)

Solution (Scatter plot)

```
import numpy as np
import seaborn as sns

x = np.array([53, 59, 65, 78, 82, 79, 84, 77])
y = np.array([156, 167, 173, 178, 189, 194, 171, 185])

sns.scatterplot(x, y);
```

Solution (Regression model)

```
import numpy as np
from scipy import stats

x = np.array([53, 59, 65, 78, 82, 79, 84, 77])
y = np.array([156, 167, 173, 178, 189, 194, 171, 185])

slope, intercept, r, p, std_err = stats.linregress(x, y)
print(“The regression model is y = ”, intercept, “+”, slope, “*weights”
```

Solution (MSE)

```
import numpy as np

polynomialorder = 1
model = np.polyfit(x, y, polynomialorder)
modelpredictor = np.polyval(model, x)
absError = modelpredictor - y
SE = np.square(absError)
MSE = np.mean(SE)
print(MSE)
```