Assignment on Linear Regression: The following table shows the results of a recently conducted study on the correlation of the number of hours spent driving with the risk of developing acute backache. Find the equation of the best fit line for this data.

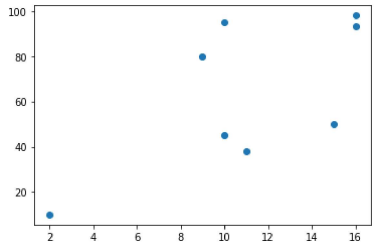
```
import pandas as pd
import numpy as np
```

```
X_driving_hours = [10, 9, 2, 15, 10, 16, 11, 16]
y_risk_score = [95, 80, 10, 50, 45, 98, 38, 93]
```

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

```
plt.plot(X_driving_hours, y_risk_score, "o")
```

```
[<matplotlib.lines.Line2D at 0x7f13675a8050>]
```



1. Finding coefficients b0, b1 that satisfy the equation y = b1x + b0 for given data

```
# helper functions
def mean(values):
    return sum(values) / float(len(values))

def variance(values, mean):
    return sum([(x-mean)**2 for x in values])

def covariance(x, mean_x, y, mean_y):
    covar = 0.0
    for i in range(len(x)):
        covar += (x[i] - mean_x) * (y[i] - mean_y)
    return covar
```

```
mean(X_driving_hours)
covariance(X_driving_hours,mean(X_driving_hours),y_risk_score,mean(y_risk_score))
```

```
701.375
```

```
def find_coefficients(x, y):
    x_mean, y_mean = mean(x), mean(y)
    b1 = covariance(x, x_mean, y, y_mean) / variance(x, x_mean)
    b0 = y_mean - b1 * x_mean
    return [b0, b1]
```

```
b0, b1 = find_coefficients(X_driving_hours, y_risk_score)
print('Coefficients: B0=%.3f, B1=%.3f' % (b0, b1))
```

```
Coefficients: B0=12.585, B1=4.588
```

The equation for the given data (in the form y = b0 + b1x) **y = 12.585 + 4.588x**

```
from sklearn.linear_model import LinearRegression
X = np.array(X_driving_hours).reshape(-1, 1)
y = np.array(y_risk_score).reshape(-1, 1)
lm = LinearRegression().fit(X, y)
lm.coef_
```

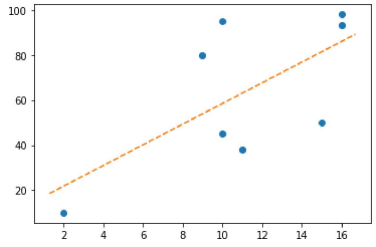
```
array([[4.58789861]])
```

```
lm.intercept_
```

```
array([12.58462796])
```

```
def abline(slope, intercept):
    axes = plt.gca()
    x_vals = np.array(axes.get_xlim())
    y_vals = intercept + slope * x_vals
    plt.plot(x_vals, y_vals, '--')
```

```
plt.plot(X_driving_hours, y_risk_score, "o")
abline(b1, b0)
```



2. Prediction

```
def linear_regression(X_train, y_train, X_test):
    predictions = list()
    b0, b1 = find_coefficients(X_train, y_train)
    for row in X_test:
        yhat = b0 + b1 * row
        predictions.append(yhat)
    return predictions


n = int(input("Enter number of predictions required: "))
results = []
for i in range(0, n):
    ele = int(input())
    results.append(ele)

predictions = linear_regression(X_driving_hours, y_risk_score, results)
print(predictions)
```

```
 Enter number of predictions required: 3
 14
 19
 5
 [76.81520850367949, 99.75470155355683, 35.52412101390024]
```

```
# Calculate Root Mean Squared Error
from math import sqrt
def calc_rmse(actual, predicted):
    sum_err = 0.0
    for i in range(len(actual)):
        pred_err = predicted[i] - actual[i]
        sum_err += (pred_err ** 2)
    mean_err = sum_err / float(len(actual))
    return sqrt(mean_err)
```