Q1. Using the boston dataset from sklearn.datasets module, perform the following tasks:
i. Load the data and show its description
ii. Load the data in a pandas dataframe
iii. To the dataframe, add the "Median value of owner-occupied homes in $1000's" from boston.target as a new column with name "MEDV"
iv. Determine the self correlation values rounded to 3 decimal places
v. Use the two features: one with Maximum correlation value and "CRIM" to create the dataset for Linear Regression model
vi. Split the data in 70:30 ratio for training and testing
vii. Using sklearn.linear_model module, load the model, train it using train data and get the predictions on test data
viii. Determine the Root Mean Square Error (RMSE) value using numpy and mean_squared_error from sklearn.metrics module, and the R2 score
ix. Use Ridge Regression on the same features with same train:test ratio using
a. alpha = 1.5 and
b. alpha = 3.5 . Determine the RMSE and R2 scores in each case.
x. Use Lasso Regression on the same features with same train:test ratio using
a. alpha = 0.5 and
b. Alpha = 1 . Determine the RMSE and R2 scores in each case.

```
#i
from sklearn.datasets import load_boston
boston = load_boston()
print(boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwi
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.
```

```
This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegie Mellc

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that addres
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data anc
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceec
```

```
#ii
import pandas as pd
ds = pd.DataFrame(boston.data,columns=boston.feature_names)
```

```
#iii
ds['MEDV'] = boston.target
```

```
#iv
pd.DataFrame(ds.corr().round(3))
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRA
CRIM	1.000	-0.200	0.407	-0.056	0.421	-0.219	0.353	-0.380	0.626	0.583	0

```
#v
x=ds['RM']
y=ds['CRIM']
pd.DataFrame([x,y])
```

	0	1	2	3	4	5	6	7	8	
RM	6.57500	6.42100	7.18500	6.99800	7.14700	6.43000	6.01200	6.17200	5.63100	6
CRIM	0.00632	0.02731	0.02729	0.03237	0.06905	0.02985	0.08829	0.14455	0.21124	0

2 rows × 506 columns

```
#vi
from sklearn.model_selection import train_test_split
x=pd.DataFrame(x)
y=pd.DataFrame(y)
x_train_1,x_test_1,y_train_1,y_test_1=train_test_split(x,y,test_size=0.3)
```

```
#vii
from sklearn.linear_model import LinearRegression
train_model=LinearRegression()
train_model.fit(x_train_1,y_train_1)
x_pred=train_model.predict(x_test_1)
y_pred=train_model.predict(y_test_1)
```

```
#viii
import numpy as np
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test_1,y_pred))
```

```
24.868693199028918
```

```
#ix a.
from sklearn.linear_model import Ridge
ridge_reg=Ridge(alpha=1.5)
```

```
#ix b.
ridge_reg2=Ridge(alpha=3.5)
ridge_reg2.fit(x_train_1,y_train_1)
print(ridge_reg2.fit(x_train_1,y_train_1))
yTestPredict=ridge_reg2.predict(x_test_1)
np.sqrt(mean_squared_error(y_test_1,yTestPredict))
```

```
Ridge(alpha=3.5, copy_X=True, fit_intercept=True, max_iter=None,
      normalize=False, random_state=None, solver='auto', tol=0.001)
5.980576670029153
```

```
#x a.
from sklearn.linear_model import Lasso
lasso_reg=Lasso(alpha=0.1)


#x b.
lasso_reg1=Lasso(alpha=1)
lasso_reg1.fit(x_train_1,y_train_1)
print(lasso_reg1.fit(x_train_1,y_train_1))
yTestPredict=lasso_reg1.predict(x_test_1)
np.sqrt(mean_squared_error(y_test_1,yTestPredict))
```

```
Lasso(alpha=1, copy_X=True, fit_intercept=True, max_iter=1000, normalize=False,
      positive=False, precompute=False, random_state=None, selection='cyclic',
      tol=0.0001, warm_start=False)
6.205856008408799
```