In [ ]:

In [ ]:

1. Now, let's call the particular libraries :

page 74 book 4

In [228]:

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

2. Now Read the DataSet :

page 74 book 4

In [61]:
```
df = pd.read_csv("DataS/USA_Housing.csv")
df
```

Out[61]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
...	...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

5000 rows × 7 columns

3. Now, TAKING 'TOP 5' AND 'TAIL 5' RECORDS :

PAGE 75 BOOK 4

In [62]:

```
df.head()
```

Out[62]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

In [63]:

```
df.tail()
```

Out[63]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

4. If we want only '1' record :

In [64]:
```
df.head(1)
```

Out[64]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...

In [65]:
```
df.tail(1)
```

Out[65]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

5. For Information : info()

page 75 book 4

In [66]:

```
# Here, Range Index : 5000 entries, (total 7 columns),
# and There are 'no nulls', Data is Fully Perfect.


df.info()
```

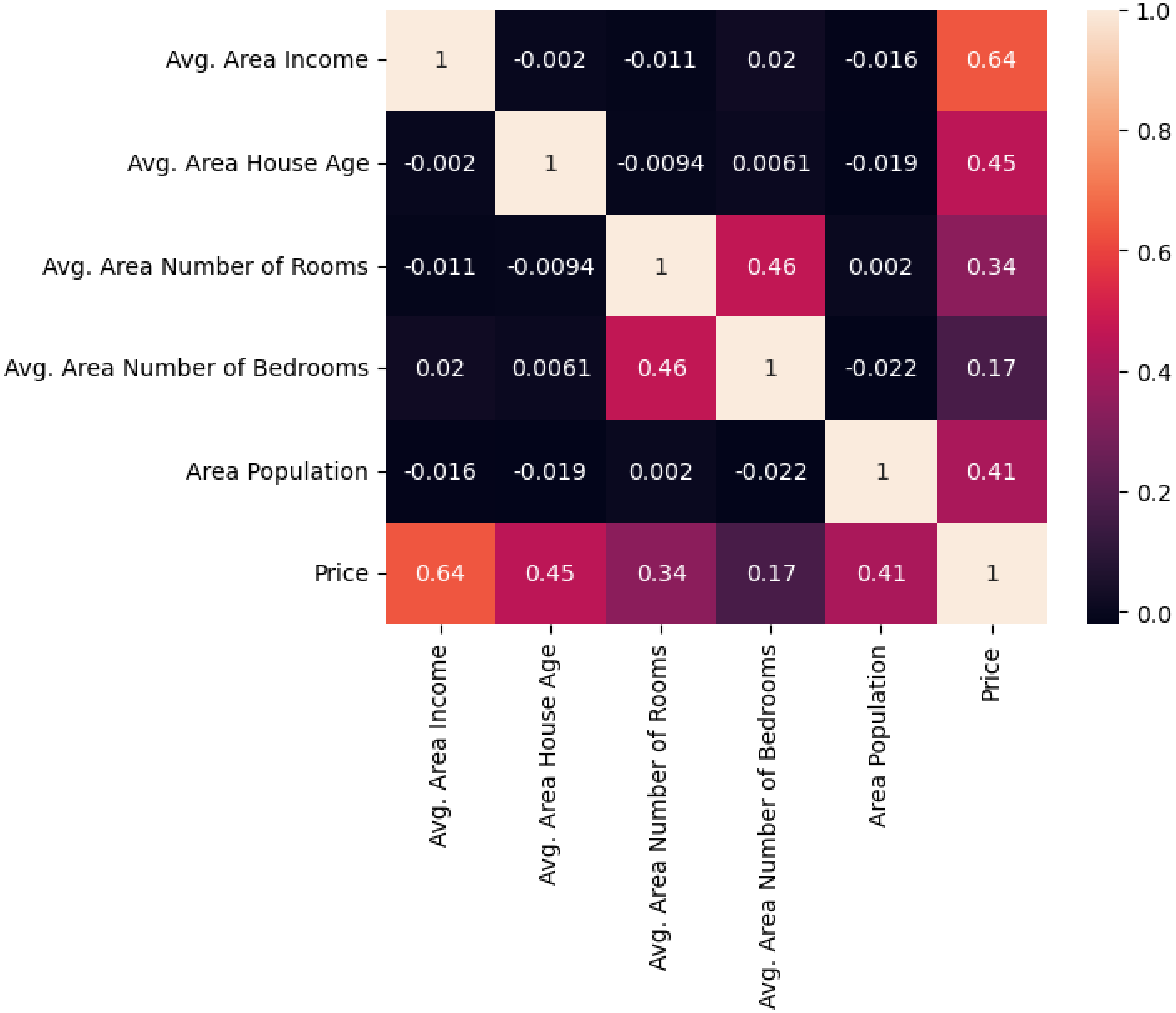
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

6. Now import 'Seaborn' :

page 75 book 4

In [67]:
```
import seaborn as sns
sns.heatmap(df.corr(), annot = True)
```

Out[67]:
```
<AxesSubplot:>
```



7. Column Names :

page 76 book 4

In [68]:
```
# Now i want to view the column names :
```

In [69]:
```
df.columns
```

Out[69]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

8. Split the Data 'X' 'y' :

page 76 book 4

In [71]:
```
X = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]
y = df['Price']
```

9. 'sklearn library' == SCIENTIFIC LIBRARY :

PAGE 77 BOOK 4

Split X,y into 'train data' and 'test data' :

In [78]:
```
# (from sklearn.cross_validation import train_test_split) - This declaration not working :
```

In [72]:
```
from sklearn.model_selection import train_test_split
```

train_test_split(|)

In [81]:
```
# Here,
```

In [82]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

10. Calling LINEAR REGRESSION :

PAGE 78 BOOK 4

In [85]:
```
# Here, Directly Calling Linear Regression :
# L & R are Capital :

from sklearn.linear_model import LinearRegression
```

In [86]:
```
lm = LinearRegression()
```

In [87]:
```
lm.fit(X_train,y_train)
```

Out[87]:
```
LinearRegression()
```

In [88]:
```
print(lm.intercept_)
```

```
-2638142.110428682
```

In [89]:
```
lm.coef_
```

Out[89]:
```
array([2.15898874e+01, 1.66102501e+05, 1.19895936e+05, 1.90107101e+03,
       1.52315025e+01])
```

In [90]:
```
df.columns
```

Out[90]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

In [92]:
```
cdf = pd.DataFrame(lm.coef_,X.columns, columns = ['Coeff'])
```

In [94]:
```
# Here, Based on the input data, The Output data is affected . All 'X' Variables are Independent.

cdf
```

Out[94]:

	Coeff
Avg. Area Income	21.589887
Avg. Area House Age	166102.501246
Avg. Area Number of Rooms	119895.936402
Avg. Area Number of Bedrooms	1901.071012
Area Population	15.231503

11. How to Call the Data Set :

page 81 book 4

In [ ]:
```
# NOTE :

# 'boston' - Boston DataSet is an an a 'Inbuilt Data Set', Which is in the form of an a 'Dictionary'.
# first we need to make ''
```

In [96]:
```
from sklearn.datasets import load_boston
```

In [99]:
```
df = load_boston()
```

In [101]:
```
# Here, 'data'- is an 'X' data, and 'target' - is an a 'y' data.
# 'feature_names' - are "labels of 'X' and 'y' ".
# Here, Data is in a Dictionary Pattern. Now this is Perfect Data Set :

df.keys()
```

Out[101]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'data_module'])
```

12. Linear Regression applying on "Sales based on - TV Marketing Budget" :

page 83 book 4

In [132]:
```
import pandas as pd
import numpy as np
```

In [141]:
```
advertising = pd.read_csv("DataS/tvmarketing.csv")
advertising
```

Out[141]:
```
        TV  Sales
  0  230.1   22.1
  1   44.5   10.4
  2   17.2    9.3
  3  151.5   18.5
  4  180.8   12.9
...    ...    ...
195   38.2    7.6
196   94.2    9.7
197  177.0   12.8
198  283.6   25.5
199  232.1   13.4

200 rows × 2 columns
```

In [142]:
```
advertising.head()
```

Out[142]:
```
      TV  Sales
0  230.1   22.1
1   44.5   10.4
2   17.2    9.3
3  151.5   18.5
4  180.8   12.9
```

In [143]:
```
advertising.tail()
```

Out[143]:

	TV	Sales
195	38.2	7.6
196	94.2	9.7
197	177.0	12.8
198	283.6	25.5
199	232.1	13.4

In [144]:
```
# Here, We don't have any Nulls(NaN's) : So, the Data is Very Perfect :

advertising.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   TV      200 non-null    float64
 1   Sales   200 non-null    float64
dtypes: float64(2)
memory usage: 3.2 KB
```

In [145]:
```
advertising.columns
```

Out[145]:
```
Index(['TV', 'Sales'], dtype='object')
```

describe():

page 84 book 4

In [146]:
```
advertising.describe()
```

Out[146]:

	TV	Sales
count	200.000000	200.000000
mean	147.042500	14.022500
std	85.854236	5.217457
min	0.700000	1.600000
25%	74.375000	10.375000
50%	149.750000	12.900000
75%	218.825000	17.400000
max	296.400000	27.000000

importing libraries :

In [211]:
```
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
```

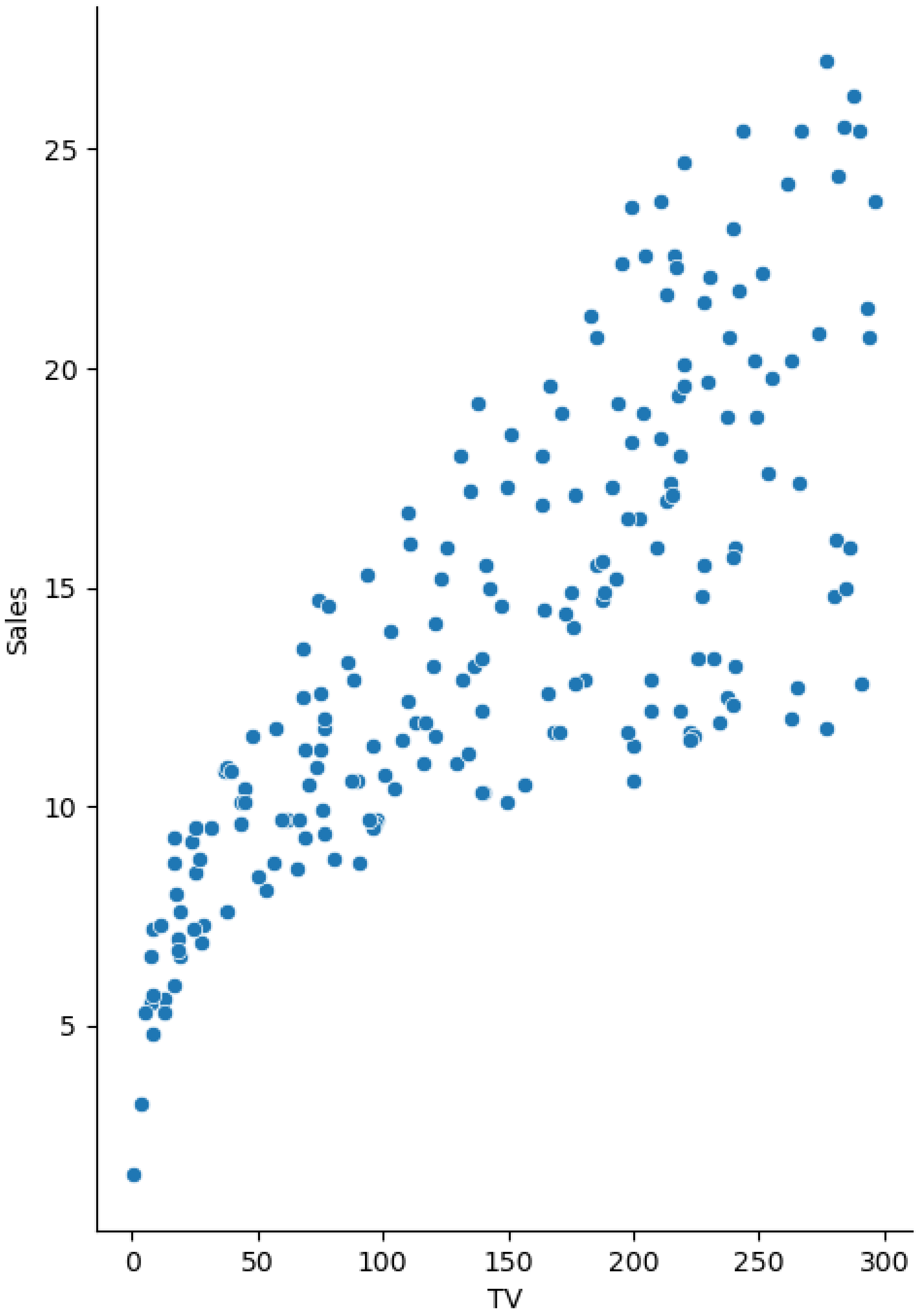
In [212]:

```
# In this particular example, we are going to observe the data and their relation,
# by using Seaborn pairlot :


sns.pairplot(advertising,x_vars = ['TV'], y_vars ='Sales',size = 7,aspect = 0.7, kind = 'scatter')
```

```
C:\Users\my pc\anaconda3\lib\site-packages\seaborn\axisgrid.py:2076: UserWarning: The `size` parameter has been rena
med to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

Out[212]:

```
<seaborn.axisgrid.PairGrid at 0x2402e1a8e20>
```


Sales
25
20
15
10
5
0
50
100
150
200
250
300
TV

Now, Split Only 'TV' Data :

page 86 book 4

In [213]:

```
# Capital 'X' :

X = advertising['TV']
X.head()
```

Out[213]:

```
0    230.1
1     44.5
2     17.2
3    151.5
4    180.8
Name: TV, dtype: float64
```

'Sales Data' :

In [214]:

```
y = advertising['Sales']
y.head()
```

Out[214]:

```
0    22.1
1    10.4
2     9.3
3    18.5
4    12.9
Name: Sales, dtype: float64
```

Now Train the Model :

In [270]:

```
from sklearn.model_selection import train_test_split
```

In [271]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

Types of an a 'X' and 'y' : 'PANDAS' CORE SERIES DATA :

In [272]:

```
print(type(X_train))
print(type(y_train))
print(type(X_test))
print(type(y_test))
```

```
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
```

'SHAPE' of an a 'X' and 'y' : Here it is '1' Dimension Data :

PAGE 88 BOOK 4

In [259]:

```
# Here, It is '1' Dimensional Data :

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(134,)
(134,)
(66,)
(66,)
```

Now, Changing X,y into '2' Dimensional Data :

By using [: , np.newaxis]

page 89 book 4

In [273]:

```
X_train = X_train[:,np.newaxis]
X_test = X_test[:,np.newaxis]
```

```
C:\Users\my pc\AppData\Local\Temp\ipykernel_20152\2733214152.py:1: FutureWarning: Support for multi-dimensional inde
xing (e.g. `obj[:, None]`) is deprecated and will be removed in a future version.  Convert to a numpy array before i
ndexing instead.
  X_train = X_train[:,np.newaxis]
C:\Users\my pc\AppData\Local\Temp\ipykernel_20152\2733214152.py:2: FutureWarning: Support for multi-dimensional inde
xing (e.g. `obj[:, None]`) is deprecated and will be removed in a future version.  Convert to a numpy array before i
ndexing instead.
  X_test = X_test[:,np.newaxis]
```

In [274]:

```
# Here, We are Converting '1' Dimensional to '2' Dimensional Data :

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(134, 1)
(134,)
(66, 1)
(66,)
```

Calling 'Linear Regression' :

page 90 book 4

In [275]:

```
from sklearn.linear_model import LinearRegression
```

In [276]:

```
# 'lr' - Some left hand Variable :

lr = LinearRegression()
```

In [277]:

```
lr.fit(X_train,y_train)
```

Out[277]:

```
LinearRegression()
```

In [278]:

```
print(lr.intercept_)
```

```
7.066582521696442
```

'Coefficient' :

In [279]:

```
print(lr.coef_)
```

```
[0.04822451]
```

'Prediction' :

In [280]:

```
x_pred = lr.predict(X_test)
x_pred
```

Out[280]:

```
array([14.94164546, 16.48965233, 21.19154233,  7.63080932, 17.69044269,
       10.68824343, 17.5216569 ,  9.47780816, 17.79171417, 15.51069472,
        8.58565468,  9.77680014, 18.37523078,  7.32699489, 13.79390206,
       15.2743946 ,  7.41862147, 16.59574625, 10.69788834, 18.51508186,
       18.13410821, 10.33620449,  8.89911401, 19.16611279, 10.39407391,
        9.64659396, 17.36251601, 13.78425716, 11.27176004,  7.47166843,
       16.70184018, 10.40371881, 16.65843812,  7.88157679, 20.60320327,
       18.55366147,  9.39582649, 20.26563168, 12.71849542,  8.39275662,
       12.66062601, 15.60232129,  9.14505902, 10.07096967, 17.86887339,
        8.90875891, 10.47123313, 14.17005326, 12.11086656, 10.74611285,
       10.83773942, 15.18759048,  7.48613578,  7.44273372, 10.7509353 ,
       13.30683448, 10.60626176, 21.03722388,  8.01178297, 16.59574625,
       20.77681152, 15.98329494, 12.51113002,  8.21432593, 21.0854484 ,
        8.00213807])
```

ACTUAL DATA VS PREDICT DATA BY 'GRAPH' :

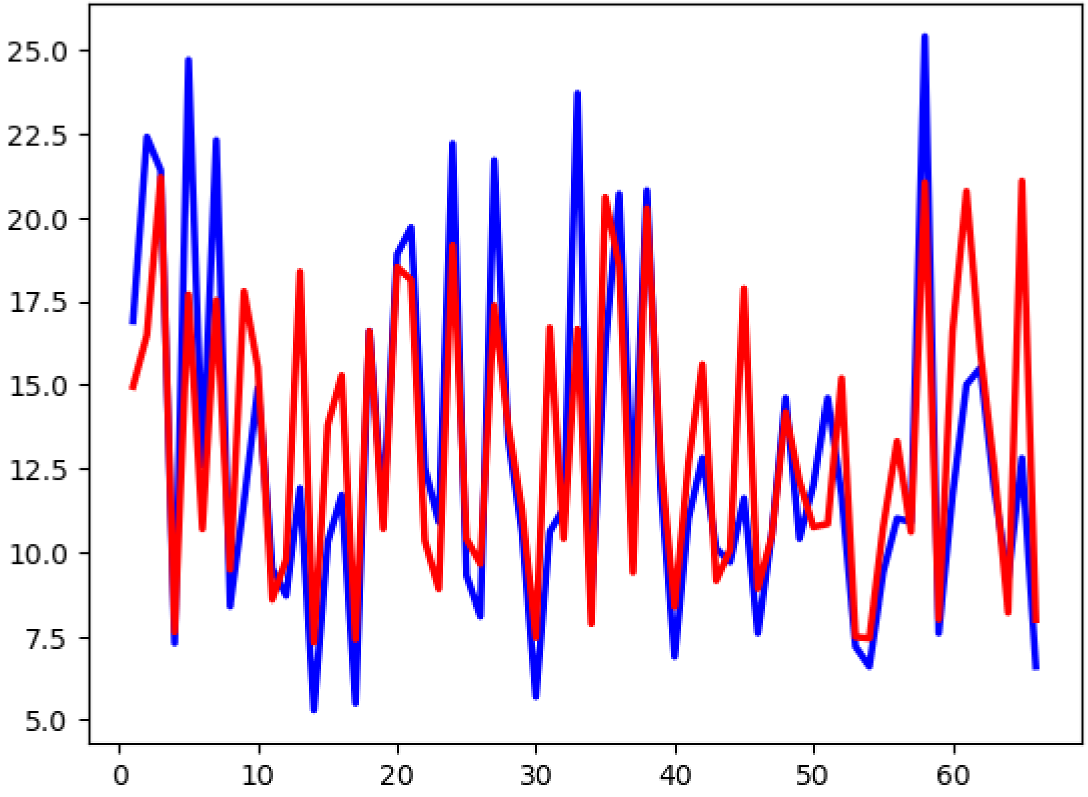
PAGE 91 BOOK 4

In [281]:

```
import matplotlib.pyplot as plt
%matplotlib inline
y_pred = lr.predict(X_test)
c = [ i for i in range(1,67,1)]
fig = plt.figure()
plt.plot(c, y_test,color = 'blue', lw = 2.5, linestyle = '-')
plt.plot(c, y_pred,color = 'r', lw = 2.5, linestyle = '-')
```

Out[281]:

```
[<matplotlib.lines.Line2D at 0x2402bca6040>]
```



NOW, 'KC HOUSING DATA' :

In [566]:
```
# Once we get this, We can do an a Graphical Representation :

data = pd.read_csv("DataS/29.kc_house_data.csv")
```

In [539]:
```
data.head()
```

Out[539]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_base
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	

5 rows × 21 columns

In [540]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64  
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64  
 9   view           21613 non-null  int64  
 10  condition      21613 non-null  int64  
 11  grade          21613 non-null  int64  
 12  sqft_above     21613 non-null  int64  
 13  sqft_basement  21613 non-null  int64  
 14  yr_built       21613 non-null  int64  
 15  yr_renovated   21613 non-null  int64  
 16  zipcode        21613 non-null  int64  
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

In [541]:

```
data.columns
```

Out[541]:

```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

In [542]:
```
data.describe()
```

Out[542]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	cond
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.00
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.40
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.65
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.00
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.00
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.00
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.00
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.00

In [543]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [544]:
```
from sklearn.model_selection import train_test_split
```

In [545]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [560]:

```
data
```

Out[560]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_l
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.0	0	0	...	8	1530	
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.0	0	0	...	8	2310	
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.0	0	0	...	7	1020	
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.0	0	0	...	8	1600	
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.0	0	0	...	7	1020	

21613 rows × 21 columns

In [547]:

```
import pandas as pd
import numpy as np
from numpy.random import randn
```

In [548]:

```
from plotly import __version__
```

In [549]:

```
import cufflinks as cf
```

In [550]:

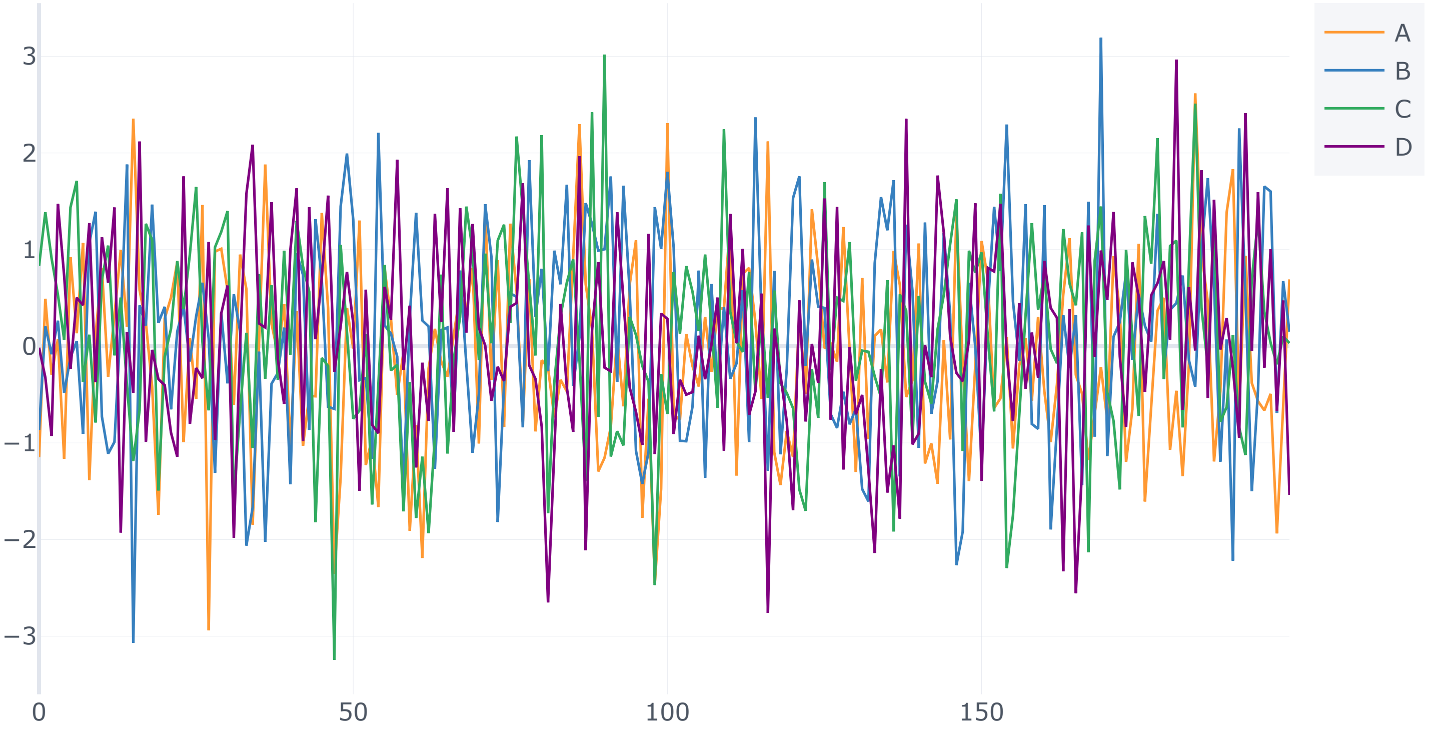
```
init_notebook_mode(connected = True)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_20152\1831628520.py in <module>
----> 1 init_notebook_mode(connected = True)

NameError: name 'init_notebook_mode' is not defined
```

In [551]:

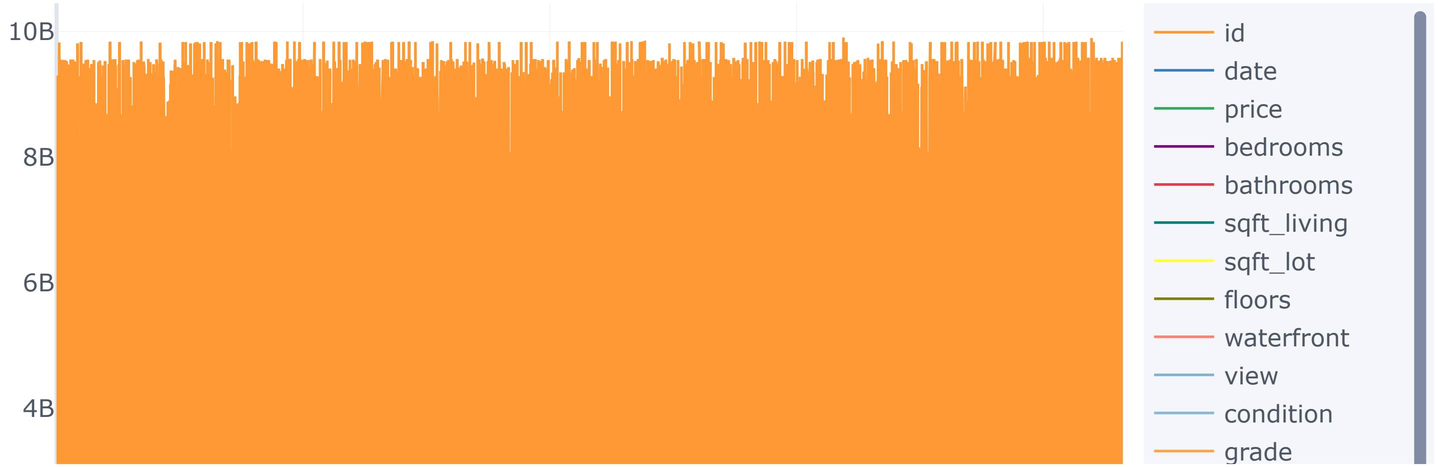
```
cf.go_offline()
```

In [556]:
```
data.iplot()
```



Export to plot.ly »

In [561]:
```
data.iplot(kind = 'scatter')
```



In [562]:
```
data = pd.DataFrame({'category':['A','B','C'], 'values': [32,43,50]})
data
```

Out[562]:

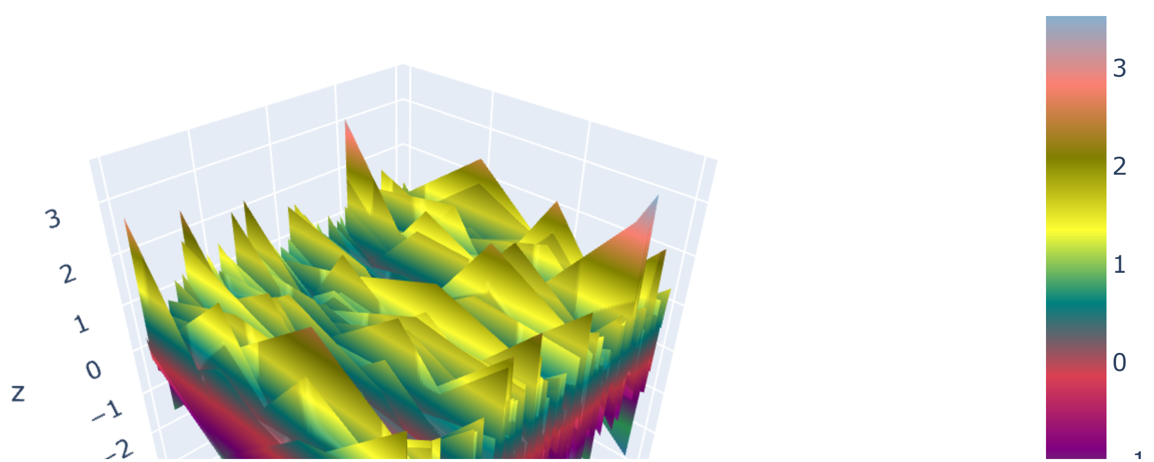
	category	values
0	A	32
1	B	43
2	C	50

In [563]:
```
data = pd.DataFrame(np.random.randn(200,4), columns = 'A B C D'.split())
data.head()
```

Out[563]:

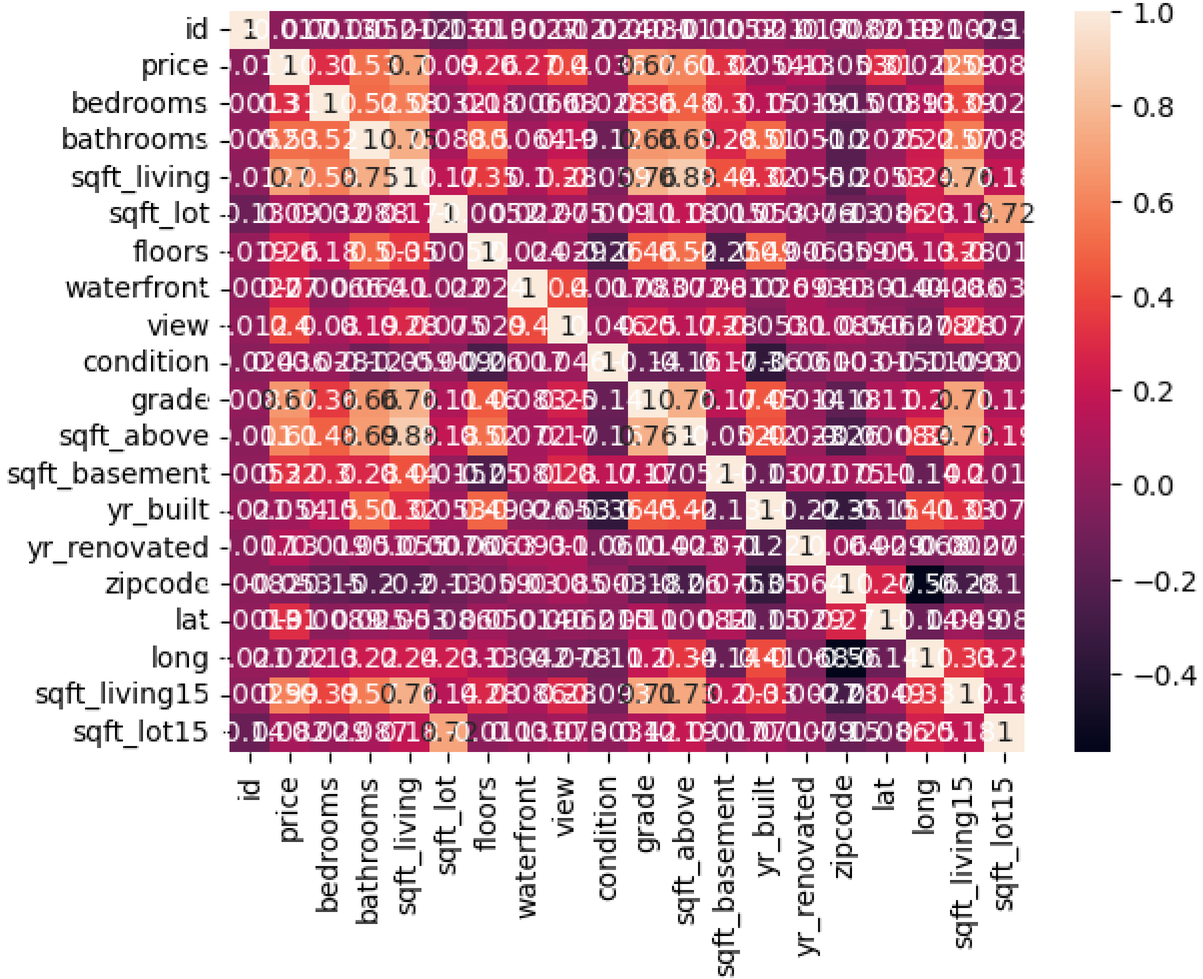
	A	B	C	D
0	1.235486	-0.853671	-0.565666	1.354380
1	0.937049	2.451492	0.738015	1.115578
2	-0.012768	1.446495	1.210544	0.554312
3	-0.001110	-1.967396	1.091417	0.921963
4	0.467174	-0.432148	1.741814	-0.413566

In [564]:
```
data.iplot(kind = 'surface')
```



In [567]:
```
import seaborn as sns
sns.heatmap(data.corr(), annot = True)
```

Out[567]:
```
<AxesSubplot:>
```



In [568]:

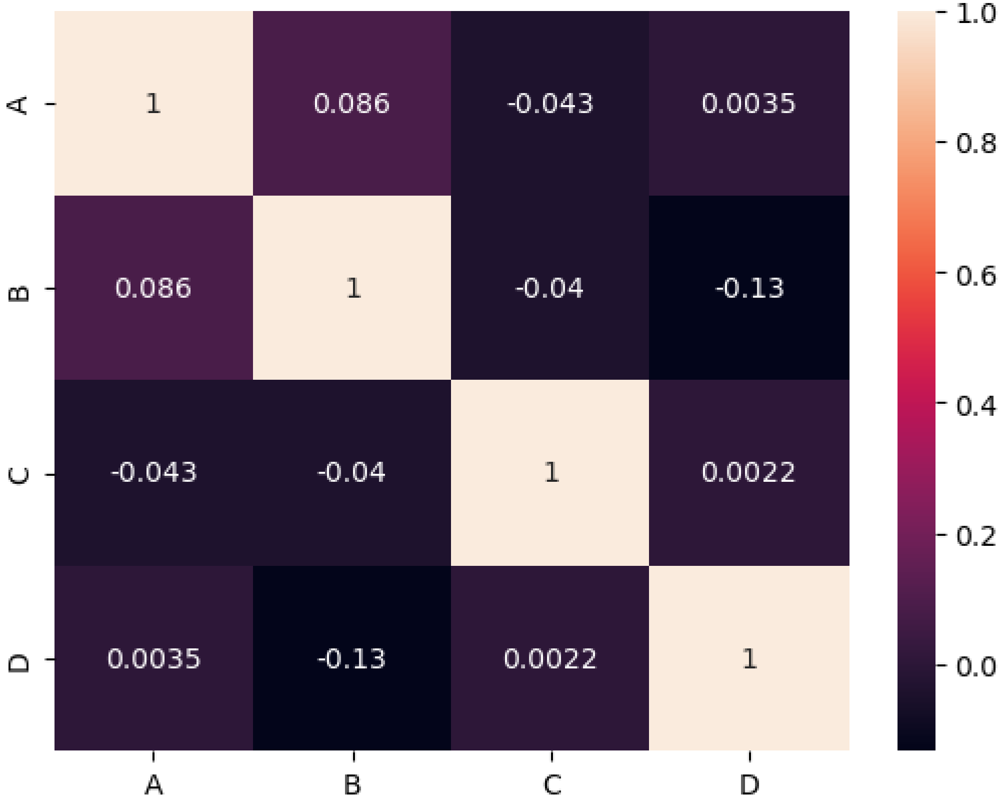
```
data = pd.DataFrame(np.random.randn(200,4), columns = 'A B C D'.split())
data.head()
```

Out[568]:

```
          A          B          C          D
0 -0.717910   0.661480   1.675569   0.541954
1  0.512613  -1.774907   0.839159  -0.899096
2  0.769996   0.052931  -0.901385  -0.510669
3 -0.214891   0.473436   0.169547   0.502355
4 -0.496329   0.223709  -1.580703  -1.568024
```

In [569]:
```
import seaborn as sns
sns.heatmap(data.corr(), annot = True)
```

Out[569]:
```
<AxesSubplot:>
```



In [570]:

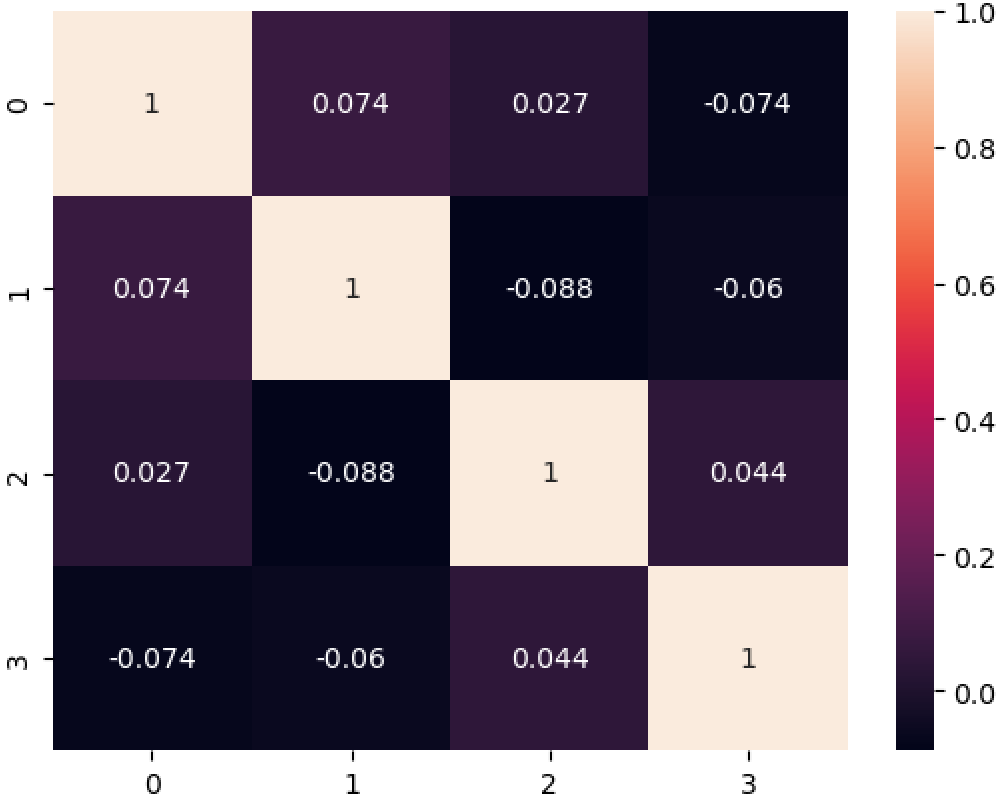
```
data = pd.DataFrame(np.random.randn(200,4))
data
```

Out[570]:

```
            0          1          2          3
  0  0.909625   1.938965  -0.386127  -1.689274
  1 -0.521288  -1.218318   1.512162   2.396411
  2 -0.119257   0.716245  -0.542812  -0.251244
  3 -0.756151  -0.059014   0.193390   0.002333
  4 -0.020800   2.585580  -1.438085  -2.467507
...       ...        ...        ...        ...
195 -0.134047  -2.277579   2.740054   0.063748
196 -0.821618  -1.169159  -0.585291  -1.503375
197  1.448589   1.940807   1.409744  -0.436775
198  0.064037  -1.050196   1.764843  -0.000398
199 -0.381303  -1.611617  -0.092121  -0.153335

200 rows × 4 columns
```

In [571]:
```
import seaborn as sns
sns.heatmap(data.corr(), annot = True)
```

Out[571]:
```
<AxesSubplot:>
```



In [572]:

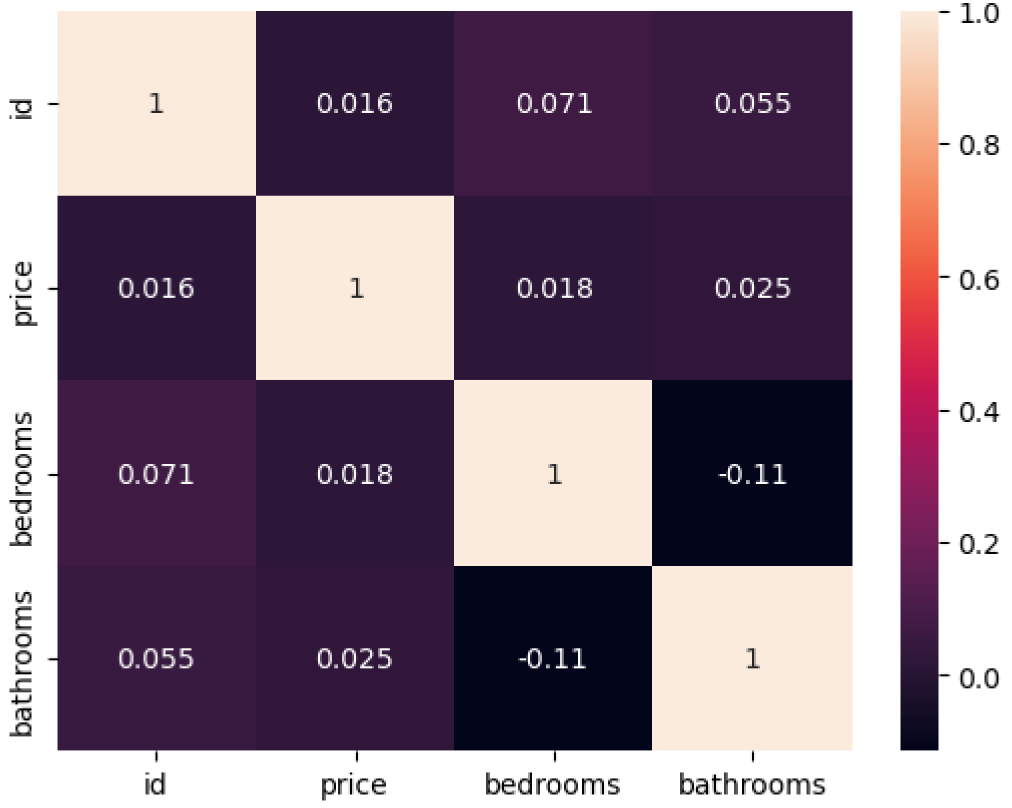
```
# just for Example :

data = pd.DataFrame(np.random.randn(200,4), columns = 'id price bedrooms bathrooms'.split())
data.head()
```

Out[572]:

	id	price	bedrooms	bathrooms
0	1.700618	-0.148633	-0.480701	0.208101
1	0.523194	-0.912204	0.267056	0.278257
2	-0.070691	0.184255	0.780837	-0.362575
3	-0.053029	-0.302490	-0.339402	-1.519880
4	1.410595	0.631424	-1.381332	-0.213058

In [573]:
```
import seaborn as sns
sns.heatmap(data.corr(), annot = True)
```

Out[573]:
```
<AxesSubplot:>
```



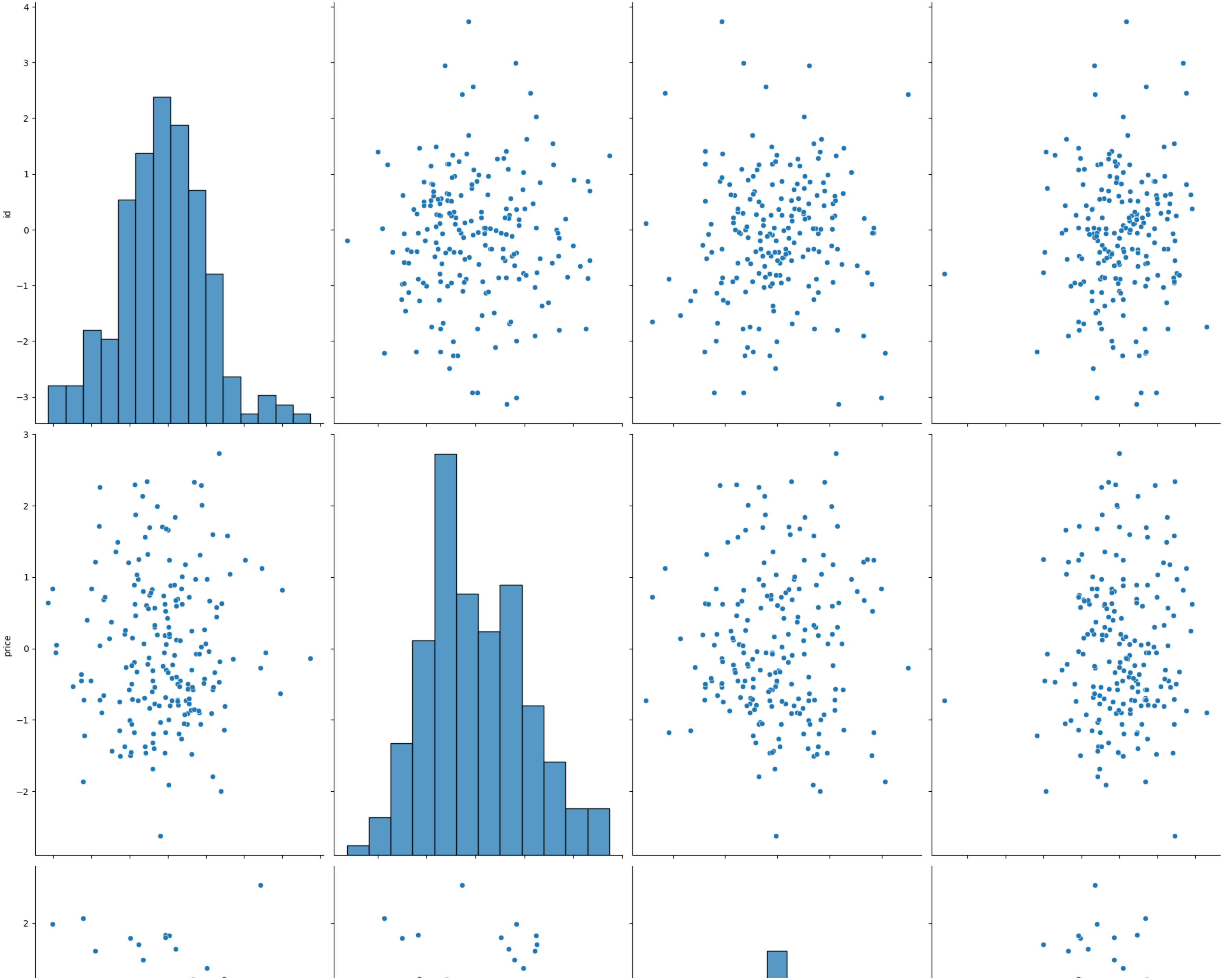
In [574]:

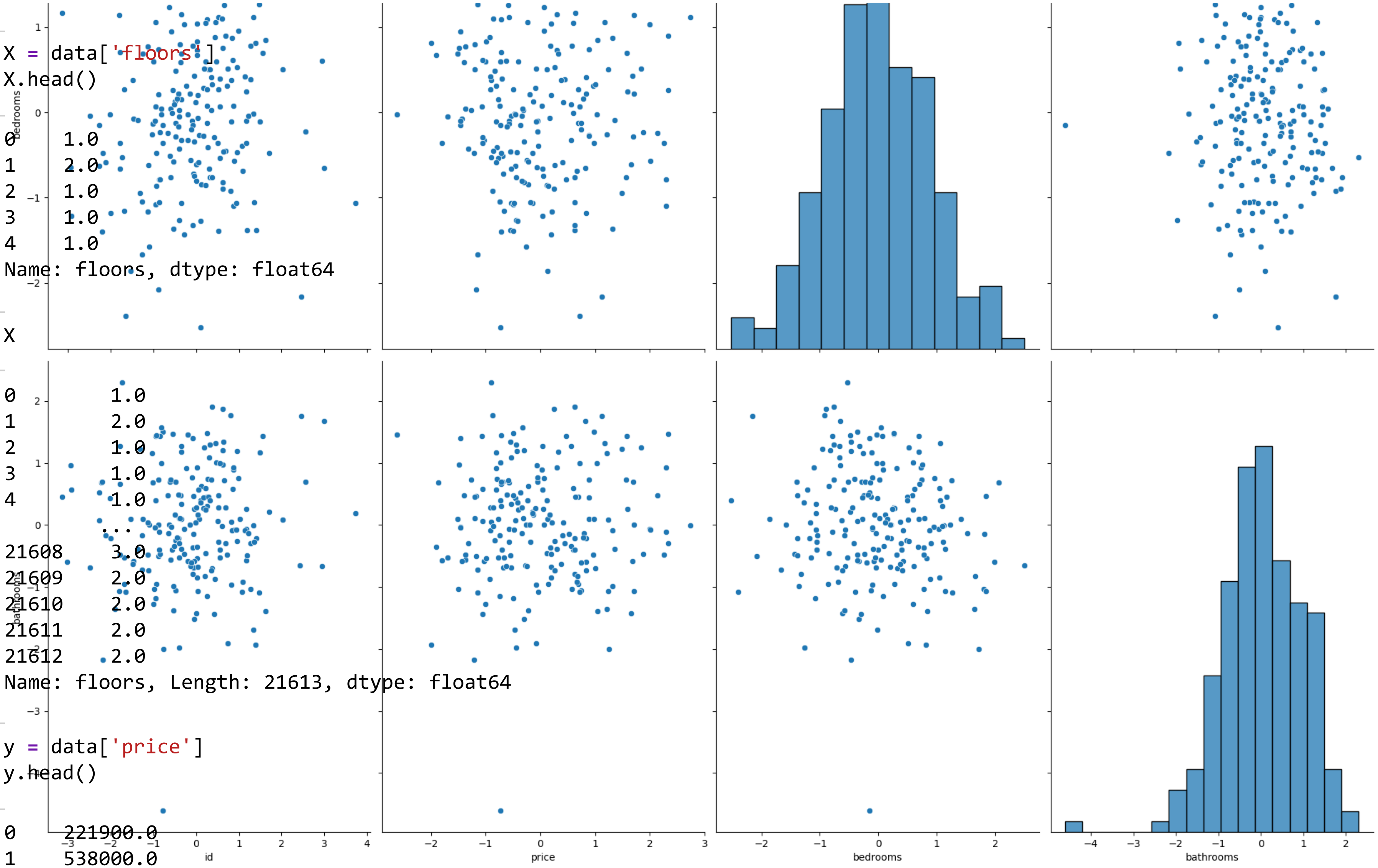
```
sns.pairplot(data, size = 7,aspect = 0.7, kind = 'scatter')
```

```
C:\Users\my pc\anaconda3\lib\site-packages\seaborn\axisgrid.py:2076: UserWarning:

The `size` parameter has been renamed to `height`; please update your code.
```

Out[574]:

```
<seaborn.axisgrid.PairGrid at 0x2403c9fb940>
```


id
4
3
2
1
0
−1
−2
−3
price
3
2
1
0
−1
−2
2



```
In [467]: X = data['floors']
          X.head()

Out[467]: 0    1.0
          1    2.0
          2    1.0
          3    1.0
          4    1.0
          Name: floors, dtype: float64
```

```
In [399]: X

Out[399]: 0        1.0
          1        2.0
          2        1.0
          3        1.0
          4        1.0
                  ...
          21608    3.0
          21609    2.0
          21610    2.0
          21611    2.0
          21612    2.0
          Name: floors, Length: 21613, dtype: float64
```

```
In [468]: y = data['price']
          y.head()

Out[468]: 0    221900.0
          1    538000.0
          2    180000.0
          3    604000.0
          4    510000.0
          Name: price, dtype: float64
```

In [441]:
```
y
```

Out[441]:
```
0        221900.0
1        538000.0
2        180000.0
3        604000.0
4        510000.0
           ...
21608    360000.0
21609    400000.0
21610    402101.0
21611    400000.0
21612    325000.0
Name: price, Length: 21613, dtype: float64
```

In [575]:
```
from sklearn.model_selection import train_test_split
```

In [576]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [577]:
```
print(type(X_train))
print(type(y_train))
print(type(X_test))
print(type(y_test))
```

```
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
```

In [578]:

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(14480,)
(14480,)
(7133,)
(7133,)
```

In [579]:

```
X_train = X_train[:,np.newaxis]
X_test = X_test[:,np.newaxis]
```

```
C:\Users\my pc\AppData\Local\Temp\ipykernel_20152\2733214152.py:1: FutureWarning:

Support for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a future version.
Convert to a numpy array before indexing instead.

C:\Users\my pc\AppData\Local\Temp\ipykernel_20152\2733214152.py:2: FutureWarning:

Support for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a future version.
Convert to a numpy array before indexing instead.
```

In [580]:

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(14480, 1)
(14480,)
(7133, 1)
(7133,)
```

In [581]:

```
from sklearn.linear_model import LinearRegression
```

```
In [582]: ln = LinearRegression()
```

```
In [583]: ln.fit(X_train,y_train)
```

```
Out[583]: LinearRegression()
```

```
In [584]: ln.intercept_
```

```
Out[584]: 285395.0290051142
```

```
In [585]: print(ln.intercept_)
```

```
          285395.0290051142
```

```
In [586]: print(ln.coef_)
```

```
          [168371.20436773]
```

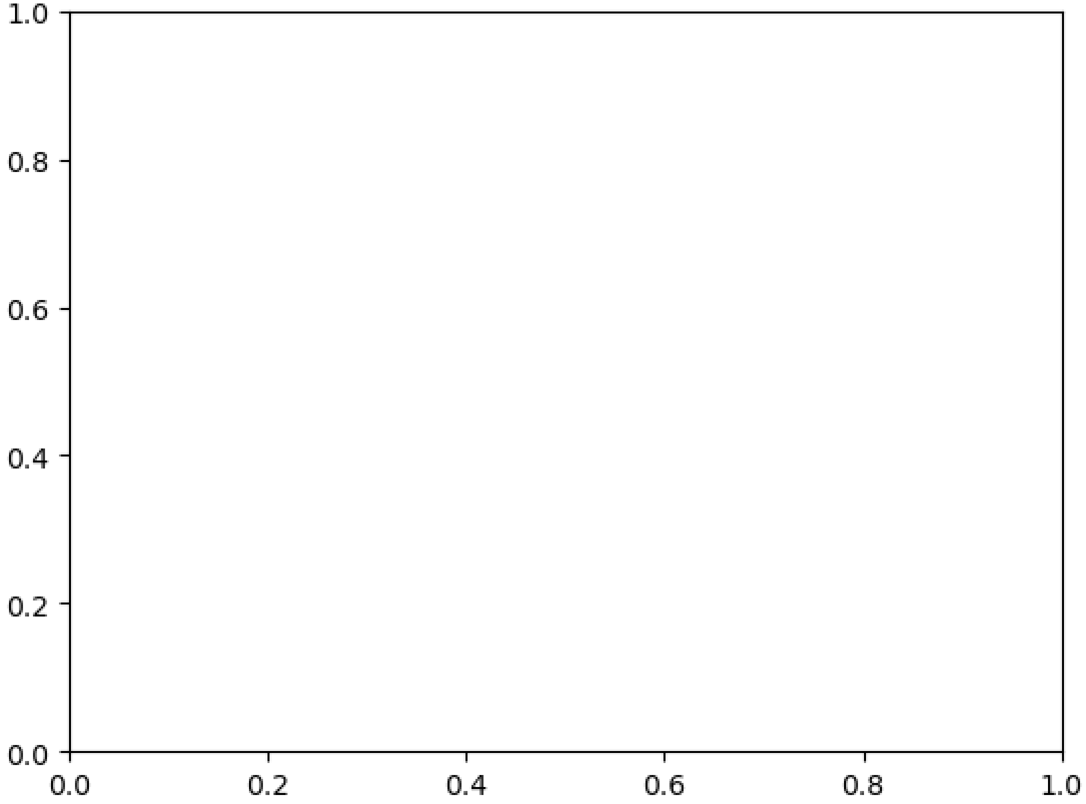
```
In [587]: x_pred = ln.predict(X_test)
          x_pred
```

```
Out[587]: array([622137.43774057, 453766.23337284, 622137.43774057, ...,
                 622137.43774057, 790508.6421083 , 453766.23337284])
```

In [588]:

```
import matplotlib.pyplot as plt
%matplotlib inline
y_pred = ln.predict(X_test)
c = [ i for i in range(1,67,1)]
fig = plt.figure()
plt.plot(c, y_test,color = 'blue', lw = 2.5, linestyle = '-')
plt.plot(c, y_pred,color = 'r', lw = 2.5, linestyle = '-')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_20152\3983697950.py in <module>
      4 c = [ i for i in range(1,67,1)]
      5 fig = plt.figure()
----> 6 plt.plot(c, y_test,color = 'blue', lw = 2.5, linestyle = '-')
      7 plt.plot(c, y_pred,color = 'r', lw = 2.5, linestyle = '-')

~\anaconda3\lib\site-packages\matplotlib\pyplot.py in plot(scalex, scaley, data, *args, **kwargs)
   2767 @_copy_docstring_and_deprecators(Axes.plot)
   2768 def plot(*args, scalex=True, scaley=True, data=None, **kwargs):
-> 2769     return gca().plot(
   2770         *args, scalex=scalex, scaley=scaley,
   2771         **({"data": data} if data is not None else {}), **kwargs)

~\anaconda3\lib\site-packages\matplotlib\axes\_axes.py in plot(self, scalex, scaley, data, *args, **kwargs)
   1630         """
   1631         kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1632         lines = [*self._get_lines(*args, data=data, **kwargs)]
   1633         for line in lines:
   1634             self.add_line(line)

~\anaconda3\lib\site-packages\matplotlib\axes\_base.py in __call__(self, data, *args, **kwargs)
    310                 this += args[0],
    311                 args = args[1:]
--> 312             yield from self._plot_args(this, kwargs)
    313 
    314     def get_next_color(self):

~\anaconda3\lib\site-packages\matplotlib\axes\_base.py in _plot_args(self, tup, kwargs, return_kwargs)
    496 
    497         if x.shape[0] != y.shape[0]:
--> 498             raise ValueError(f"x and y must have same first dimension, but "
    499                              f"have shapes {x.shape} and {y.shape}")
    500         if x.ndim > 2 or y.ndim > 2:

ValueError: x and y must have same first dimension, but have shapes (66,) and (7133,)
```



```
In [ ]:
```