___Apple Stock Price Prediction_

importing necessary libraries

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

reading the data file

In [7]:
```
df=pd.read_csv("AAPL.csv")
```

In [10]:
```
df.head()
```

Out[10]:

	Unnamed: 0	symbol	date	close	high	low	open	volume	adjClose	a
0	0	AAPL	2015-05-27 00:00:00+00:00	132.045	132.260	130.05	130.34	45833246	121.682558	121
1	1	AAPL	2015-05-28 00:00:00+00:00	131.780	131.950	131.10	131.86	30733309	121.438354	121
2	2	AAPL	2015-05-29 00:00:00+00:00	130.280	131.450	129.90	131.23	50884452	120.056069	121
3	3	AAPL	2015-06-01 00:00:00+00:00	130.535	131.390	130.05	131.20	32112797	120.291057	121
4	4	AAPL	2015-06-02 00:00:00+00:00	129.960	130.655	129.32	129.86	33667627	119.761181	12C

In [11]:
```
df.tail()
```

Out[11]:

	Unnamed: 0	symbol	date	close	high	low	open	volume	adjClose	ac
1253	1253	AAPL	2020-05-18 00:00:00+00:00	314.96	316.50	310.3241	313.17	33843125	314.96	
1254	1254	AAPL	2020-05-19 00:00:00+00:00	313.14	318.52	313.0100	315.03	25432385	313.14	
1255	1255	AAPL	2020-05-20 00:00:00+00:00	319.23	319.52	316.2000	316.68	27876215	319.23	
1256	1256	AAPL	2020-05-21 00:00:00+00:00	316.85	320.89	315.8700	318.66	25672211	316.85	
1257	1257	AAPL	2020-05-22 00:00:00+00:00	318.89	319.23	315.3500	315.77	20450754	318.89	

So the data shows, initial date as 2015-05-27 and last date as 2020-05-22

In [14]:
```
df1=df.reset_index()["close"]
```

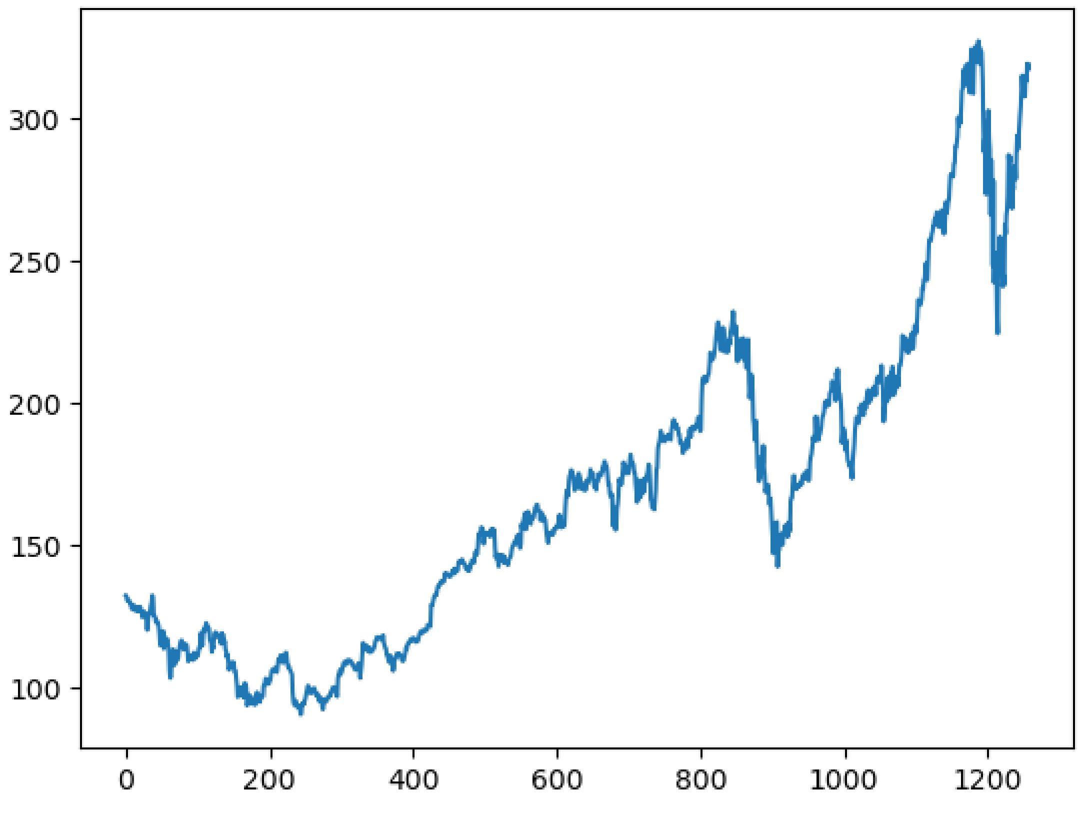
In [15]:
```
df1
```

Out[15]:
```
0       132.045
1       131.780
2       130.280
3       130.535
4       129.960
         ...
1253    314.960
1254    313.140
1255    319.230
1256    316.850
1257    318.890
Name: close, Length: 1258, dtype: float64
```

Plotting the "close" attribute pattern

In [16]:
```
plt.plot(df1)
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x29a211f1310>]
```



scaling down all the values between 0 to 1

In [17]:
```
from sklearn.preprocessing import MinMaxScaler
```

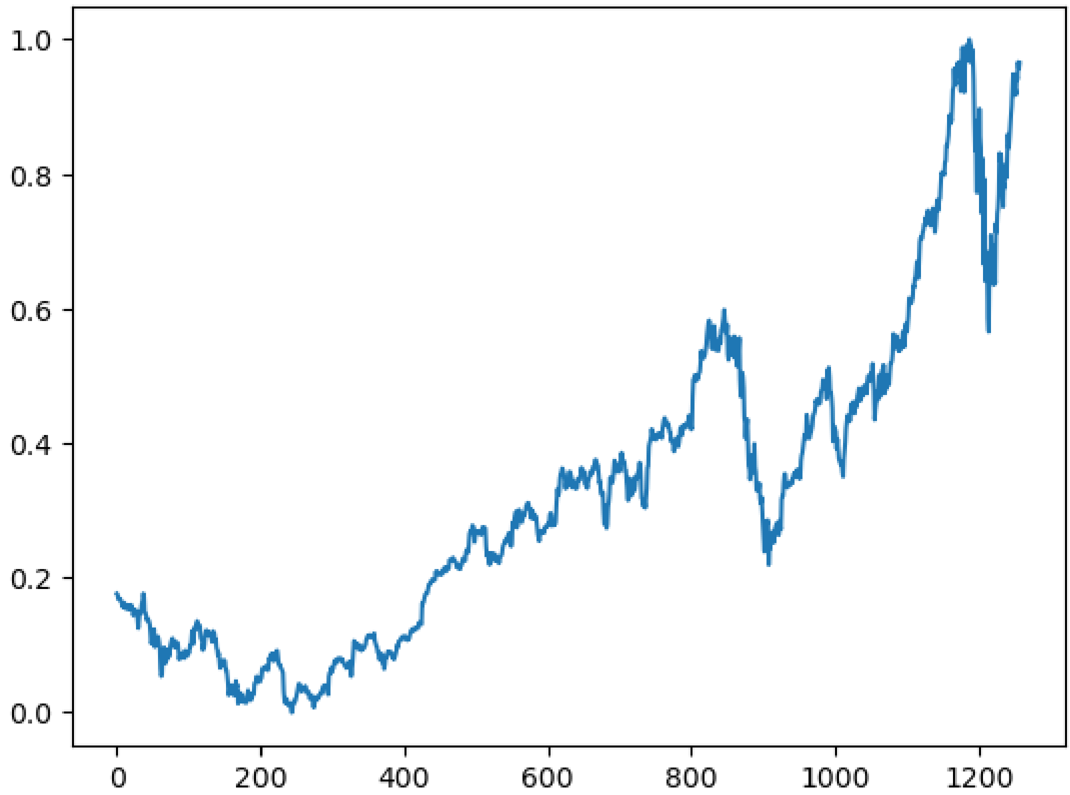
In [18]:
```
mmscaler=MinMaxScaler()
```

In [19]:
```
df1=mmscaler.fit_transform(np.array(df1).reshape(-1,1))
df1
```

Out[19]:
```
array([[0.17607447],
       [0.17495567],
       [0.16862282],
       ...,
       [0.96635143],
       [0.9563033 ],
       [0.96491598]])
```

In [20]:
```
plt.plot(df1)
```

Out[20]:
```
[<matplotlib.lines.Line2D at 0x29a21c7aa00>]
```



dividing the data into two parts initially

In [24]:
```
training_size=int(len(df1)*0.65)
test_size=len(df1)-training_size
```

In [35]:
```
training_size,test_size
```

Out[35]:
```
(817, 441)
```

In [33]:
```
training_dataset,testing_dataset=df1[:training_size],df1[training_size:len(df1)]
```

In [34]:
```
training_dataset.shape,testing_dataset.shape
```

Out[34]:
```
((817, 1), (441, 1))
```

defined a function which will give arrays according to the used given training days

In [48]:
```
import numpy
def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return numpy.array(dataX), numpy.array(dataY)
```

using above function, splitted our *featured* data into train & test

In [50]:
```
time_step = 100
X_train, y_train = create_dataset(training_dataset, time_step)
X_test, ytest = create_dataset(testing_dataset, time_step)
```

In [53]:
```
X_train.shape,y_train.shape,X_test.shape,ytest.shape
```

Out[53]:
```
((716, 100), (716,), (340, 100), (340,))
```

preparing data for adding into Stacked LSTM model

In [56]:
```
X_train=X_train.reshape(X_train.shape[0],X_train.shape[1],1)
X_test=X_test.reshape(X_test.shape[0],X_test.shape[1],1)
```

In [57]:
```
X_train.shape,X_test.shape
```

Out[57]:
```
((716, 100, 1), (340, 100, 1))
```

imported necessary deep learning libraries

In [59]:
```
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

Stacked LSTM model preparation

In [60]:
```
model=Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(100,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
```

In [61]:
```
model.compile(optimizer="adam",loss="mean_squared_error")
```

In [62]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400     
                                                                 
 lstm_1 (LSTM)               (None, 100, 50)           20200     
                                                                 
 lstm_2 (LSTM)               (None, 50)                20200     
                                                                 
 dense (Dense)               (None, 1)                 51        
                                                                 
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

In [63]:
```
model.fit(X_train,y_train,epochs=100,batch_size=32,validation_data=(X_test,ytest),
```

```
Epoch 1/100
23/23 [==============================] - 12s 227ms/step - loss: 0.0094 - val_loss:
0.0043
Epoch 2/100
23/23 [==============================] - 3s 152ms/step - loss: 0.0011 - val_loss:
0.0040
Epoch 3/100
23/23 [==============================] - 3s 153ms/step - loss: 7.9071e-04 - val_lo
ss: 0.0052
Epoch 4/100
23/23 [==============================] - 4s 155ms/step - loss: 6.6859e-04 - val_lo
ss: 0.0036
Epoch 5/100
23/23 [==============================] - 3s 153ms/step - loss: 6.2450e-04 - val_lo
ss: 0.0036
Epoch 6/100
23/23 [==============================] - 4s 154ms/step - loss: 5.6268e-04 - val_lo
ss: 0.0046
Epoch 7/100
23/23 [==============================] - 4s 156ms/step - loss: 5.8390e-04 - val_lo
ss: 0.0031
Epoch 8/100
23/23 [==============================] - 3s 153ms/step - loss: 5.3337e-04 - val_lo
ss: 0.0031
Epoch 9/100
23/23 [==============================] - 4s 154ms/step - loss: 5.6353e-04 - val_lo
ss: 0.0026
Epoch 10/100
23/23 [==============================] - 4s 153ms/step - loss: 5.4198e-04 - val_lo
ss: 0.0024
Epoch 11/100
23/23 [==============================] - 3s 153ms/step - loss: 5.1113e-04 - val_lo
ss: 0.0025
Epoch 12/100
23/23 [==============================] - 3s 153ms/step - loss: 5.4891e-04 - val_lo
ss: 0.0024
Epoch 13/100
23/23 [==============================] - 3s 152ms/step - loss: 5.9291e-04 - val_lo
ss: 0.0022
Epoch 14/100
23/23 [==============================] - 3s 152ms/step - loss: 4.4310e-04 - val_lo
ss: 0.0020
Epoch 15/100
23/23 [==============================] - 3s 153ms/step - loss: 4.8440e-04 - val_lo
ss: 0.0023
Epoch 16/100
23/23 [==============================] - 4s 154ms/step - loss: 4.5040e-04 - val_lo
ss: 0.0023
Epoch 17/100
23/23 [==============================] - 3s 152ms/step - loss: 4.3766e-04 - val_lo
ss: 0.0023
Epoch 18/100
23/23 [==============================] - 4s 154ms/step - loss: 3.7756e-04 - val_lo
ss: 0.0019
Epoch 19/100
23/23 [==============================] - 3s 152ms/step - loss: 4.2423e-04 - val_lo
ss: 0.0015
Epoch 20/100
23/23 [==============================] - 4s 154ms/step - loss: 3.7579e-04 - val_lo
ss: 0.0020
Epoch 21/100
23/23 [==============================] - 3s 153ms/step - loss: 3.4317e-04 - val_lo
ss: 0.0026
Epoch 22/100
```

```
23/23 [==============================] - 3s 153ms/step - loss: 3.5865e-04 - val_lo
ss: 0.0021
Epoch 23/100
23/23 [==============================] - 4s 156ms/step - loss: 3.2300e-04 - val_lo
ss: 0.0032
Epoch 24/100
23/23 [==============================] - 3s 152ms/step - loss: 4.0658e-04 - val_lo
ss: 0.0015
Epoch 25/100
23/23 [==============================] - 3s 151ms/step - loss: 2.9848e-04 - val_lo
ss: 0.0015
Epoch 26/100
23/23 [==============================] - 4s 153ms/step - loss: 3.0781e-04 - val_lo
ss: 0.0014
Epoch 27/100
23/23 [==============================] - 3s 151ms/step - loss: 2.8197e-04 - val_lo
ss: 0.0015
Epoch 28/100
23/23 [==============================] - 3s 152ms/step - loss: 2.8556e-04 - val_lo
ss: 0.0014
Epoch 29/100
23/23 [==============================] - 3s 150ms/step - loss: 2.8697e-04 - val_lo
ss: 0.0014
Epoch 30/100
23/23 [==============================] - 3s 151ms/step - loss: 3.0364e-04 - val_lo
ss: 0.0016
Epoch 31/100
23/23 [==============================] - 3s 151ms/step - loss: 2.9811e-04 - val_lo
ss: 0.0014
Epoch 32/100
23/23 [==============================] - 3s 151ms/step - loss: 2.3977e-04 - val_lo
ss: 0.0019
Epoch 33/100
23/23 [==============================] - 3s 151ms/step - loss: 2.3856e-04 - val_lo
ss: 0.0012
Epoch 34/100
23/23 [==============================] - 3s 150ms/step - loss: 2.3249e-04 - val_lo
ss: 0.0012
Epoch 35/100
23/23 [==============================] - 4s 156ms/step - loss: 2.2331e-04 - val_lo
ss: 0.0015
Epoch 36/100
23/23 [==============================] - 3s 151ms/step - loss: 1.9758e-04 - val_lo
ss: 0.0013
Epoch 37/100
23/23 [==============================] - 4s 153ms/step - loss: 2.0376e-04 - val_lo
ss: 0.0013
Epoch 38/100
23/23 [==============================] - 3s 151ms/step - loss: 2.0963e-04 - val_lo
ss: 0.0018
Epoch 39/100
23/23 [==============================] - 3s 151ms/step - loss: 1.7451e-04 - val_lo
ss: 0.0012
Epoch 40/100
23/23 [==============================] - 3s 153ms/step - loss: 1.8054e-04 - val_lo
ss: 9.8565e-04
Epoch 41/100
23/23 [==============================] - 3s 151ms/step - loss: 1.6755e-04 - val_lo
ss: 9.8294e-04
Epoch 42/100
23/23 [==============================] - 4s 157ms/step - loss: 1.7244e-04 - val_lo
ss: 0.0010
Epoch 43/100
23/23 [==============================] - 3s 152ms/step - loss: 1.5523e-04 - val_lo
```

```
ss: 9.9591e-04
Epoch 44/100
23/23 [==============================] - 3s 150ms/step - loss: 1.8854e-04 - val_lo
ss: 9.2194e-04
Epoch 45/100
23/23 [==============================] - 3s 151ms/step - loss: 1.8331e-04 - val_lo
ss: 9.1775e-04
Epoch 46/100
23/23 [==============================] - 3s 151ms/step - loss: 1.4085e-04 - val_lo
ss: 8.6270e-04
Epoch 47/100
23/23 [==============================] - 3s 151ms/step - loss: 1.4327e-04 - val_lo
ss: 0.0017
Epoch 48/100
23/23 [==============================] - 3s 151ms/step - loss: 1.9429e-04 - val_lo
ss: 9.0809e-04
Epoch 49/100
23/23 [==============================] - 3s 151ms/step - loss: 1.4269e-04 - val_lo
ss: 8.0840e-04
Epoch 50/100
23/23 [==============================] - 3s 152ms/step - loss: 1.3244e-04 - val_lo
ss: 8.0028e-04
Epoch 51/100
23/23 [==============================] - 3s 152ms/step - loss: 1.3439e-04 - val_lo
ss: 0.0012
Epoch 52/100
23/23 [==============================] - 3s 151ms/step - loss: 1.3694e-04 - val_lo
ss: 0.0011
Epoch 53/100
23/23 [==============================] - 3s 149ms/step - loss: 1.4029e-04 - val_lo
ss: 0.0013
Epoch 54/100
23/23 [==============================] - 3s 151ms/step - loss: 1.4701e-04 - val_lo
ss: 9.8292e-04
Epoch 55/100
23/23 [==============================] - 3s 151ms/step - loss: 1.2148e-04 - val_lo
ss: 9.6539e-04
Epoch 56/100
23/23 [==============================] - 3s 151ms/step - loss: 1.2050e-04 - val_lo
ss: 8.1020e-04
Epoch 57/100
23/23 [==============================] - 3s 151ms/step - loss: 1.4495e-04 - val_lo
ss: 0.0011
Epoch 58/100
23/23 [==============================] - 3s 151ms/step - loss: 1.3703e-04 - val_lo
ss: 9.3823e-04
Epoch 59/100
23/23 [==============================] - 3s 153ms/step - loss: 1.2122e-04 - val_lo
ss: 0.0010
Epoch 60/100
23/23 [==============================] - 3s 152ms/step - loss: 1.1689e-04 - val_lo
ss: 7.8292e-04
Epoch 61/100
23/23 [==============================] - 3s 150ms/step - loss: 1.2804e-04 - val_lo
ss: 9.0188e-04
Epoch 62/100
23/23 [==============================] - 3s 151ms/step - loss: 1.2787e-04 - val_lo
ss: 0.0010
Epoch 63/100
23/23 [==============================] - 3s 152ms/step - loss: 1.0577e-04 - val_lo
ss: 8.1804e-04
Epoch 64/100
23/23 [==============================] - 3s 151ms/step - loss: 1.0817e-04 - val_lo
ss: 7.9841e-04
```

```
Epoch 65/100
23/23 [==============================] - 3s 151ms/step - loss: 1.0392e-04 - val_lo
ss: 7.9825e-04
Epoch 66/100
23/23 [==============================] - 3s 153ms/step - loss: 1.0144e-04 - val_lo
ss: 8.1503e-04
Epoch 67/100
23/23 [==============================] - 3s 151ms/step - loss: 1.1140e-04 - val_lo
ss: 9.4862e-04
Epoch 68/100
23/23 [==============================] - 4s 153ms/step - loss: 1.2225e-04 - val_lo
ss: 7.9669e-04
Epoch 69/100
23/23 [==============================] - 3s 152ms/step - loss: 1.0690e-04 - val_lo
ss: 8.0069e-04
Epoch 70/100
23/23 [==============================] - 3s 153ms/step - loss: 1.0074e-04 - val_lo
ss: 8.6567e-04
Epoch 71/100
23/23 [==============================] - 3s 150ms/step - loss: 1.0915e-04 - val_lo
ss: 7.9957e-04
Epoch 72/100
23/23 [==============================] - 3s 152ms/step - loss: 9.7625e-05 - val_lo
ss: 8.0508e-04
Epoch 73/100
23/23 [==============================] - 3s 151ms/step - loss: 1.0100e-04 - val_lo
ss: 8.1529e-04
Epoch 74/100
23/23 [==============================] - 3s 150ms/step - loss: 1.0890e-04 - val_lo
ss: 9.1021e-04
Epoch 75/100
23/23 [==============================] - 3s 151ms/step - loss: 9.7794e-05 - val_lo
ss: 8.3658e-04
Epoch 76/100
23/23 [==============================] - 4s 153ms/step - loss: 1.0941e-04 - val_lo
ss: 8.3158e-04
Epoch 77/100
23/23 [==============================] - 3s 150ms/step - loss: 9.4045e-05 - val_lo
ss: 0.0010
Epoch 78/100
23/23 [==============================] - 3s 151ms/step - loss: 9.7131e-05 - val_lo
ss: 8.8973e-04
Epoch 79/100
23/23 [==============================] - 3s 152ms/step - loss: 1.0377e-04 - val_lo
ss: 8.7121e-04
Epoch 80/100
23/23 [==============================] - 3s 151ms/step - loss: 1.0891e-04 - val_lo
ss: 8.8602e-04
Epoch 81/100
23/23 [==============================] - 3s 150ms/step - loss: 1.1190e-04 - val_lo
ss: 0.0012
Epoch 82/100
23/23 [==============================] - 3s 150ms/step - loss: 1.0823e-04 - val_lo
ss: 8.7559e-04
Epoch 83/100
23/23 [==============================] - 3s 152ms/step - loss: 9.0456e-05 - val_lo
ss: 8.8369e-04
Epoch 84/100
23/23 [==============================] - 3s 150ms/step - loss: 1.1019e-04 - val_lo
ss: 0.0010
Epoch 85/100
23/23 [==============================] - 3s 151ms/step - loss: 9.8287e-05 - val_lo
ss: 0.0011
Epoch 86/100
```

```
23/23 [==============================] - 3s 150ms/step - loss: 9.4994e-05 - val_lo
ss: 9.0211e-04
Epoch 87/100
23/23 [==============================] - 3s 151ms/step - loss: 8.7206e-05 - val_lo
ss: 9.3266e-04
Epoch 88/100
23/23 [==============================] - 3s 152ms/step - loss: 8.9925e-05 - val_lo
ss: 9.5723e-04
Epoch 89/100
23/23 [==============================] - 3s 151ms/step - loss: 9.0592e-05 - val_lo
ss: 0.0013
Epoch 90/100
23/23 [==============================] - 3s 151ms/step - loss: 9.5113e-05 - val_lo
ss: 9.4381e-04
Epoch 91/100
23/23 [==============================] - 3s 151ms/step - loss: 1.0530e-04 - val_lo
ss: 0.0012
Epoch 92/100
23/23 [==============================] - 4s 153ms/step - loss: 8.9595e-05 - val_lo
ss: 0.0012
Epoch 93/100
23/23 [==============================] - 4s 156ms/step - loss: 9.5242e-05 - val_lo
ss: 0.0010
Epoch 94/100
23/23 [==============================] - 4s 157ms/step - loss: 1.0293e-04 - val_lo
ss: 0.0010
Epoch 95/100
23/23 [==============================] - 4s 157ms/step - loss: 9.1837e-05 - val_lo
ss: 0.0011
Epoch 96/100
23/23 [==============================] - 4s 163ms/step - loss: 9.6771e-05 - val_lo
ss: 0.0011
Epoch 97/100
23/23 [==============================] - 4s 160ms/step - loss: 8.7231e-05 - val_lo
ss: 0.0011
Epoch 98/100
23/23 [==============================] - 4s 157ms/step - loss: 8.2830e-05 - val_lo
ss: 0.0013
Epoch 99/100
23/23 [==============================] - 3s 151ms/step - loss: 8.1799e-05 - val_lo
ss: 0.0012
Epoch 100/100
23/23 [==============================] - 4s 154ms/step - loss: 8.1327e-05 - val_lo
ss: 0.0016
```

Out[63]:

```
<keras.callbacks.History at 0x29a326e44f0>
```

Predicting the data and inversing it from 0 to 1 to actual value form

In [69]:

```
train_predict=model.predict(X_train)
test_predict=model.predict(X_test)
```

```
23/23 [==============================] - 3s 50ms/step
11/11 [==============================] - 1s 50ms/step
```

In [71]:

```
train_predict=mmscaler.inverse_transform(train_predict)
test_predict=mmscaler.inverse_transform(test_predict)
```

In [73]:

```
train_predict.shape
```

Out[73]:

```
(716, 1)
```

Testing Root Mean Squared Error

In [74]:
```
from sklearn.metrics import mean_squared_error
import math
rmse=math.sqrt(mean_squared_error(ytest,test_predict))
rmse
```

Out[74]:
```
233.37424217437922
```

Training RMSE value

In [82]:
```
from sklearn.metrics import mean_squared_error
import math
rmse2=math.sqrt(mean_squared_error(y_train,train_predict))
rmse2
```

Out[82]:
```
140.88441834199475
```

In [96]:
```
a=numpy.empty_like(df1)
```

In [103…
```
len(train_predict)
```

Out[103]:
```
716
```

In [104…
```
len(df1)
```

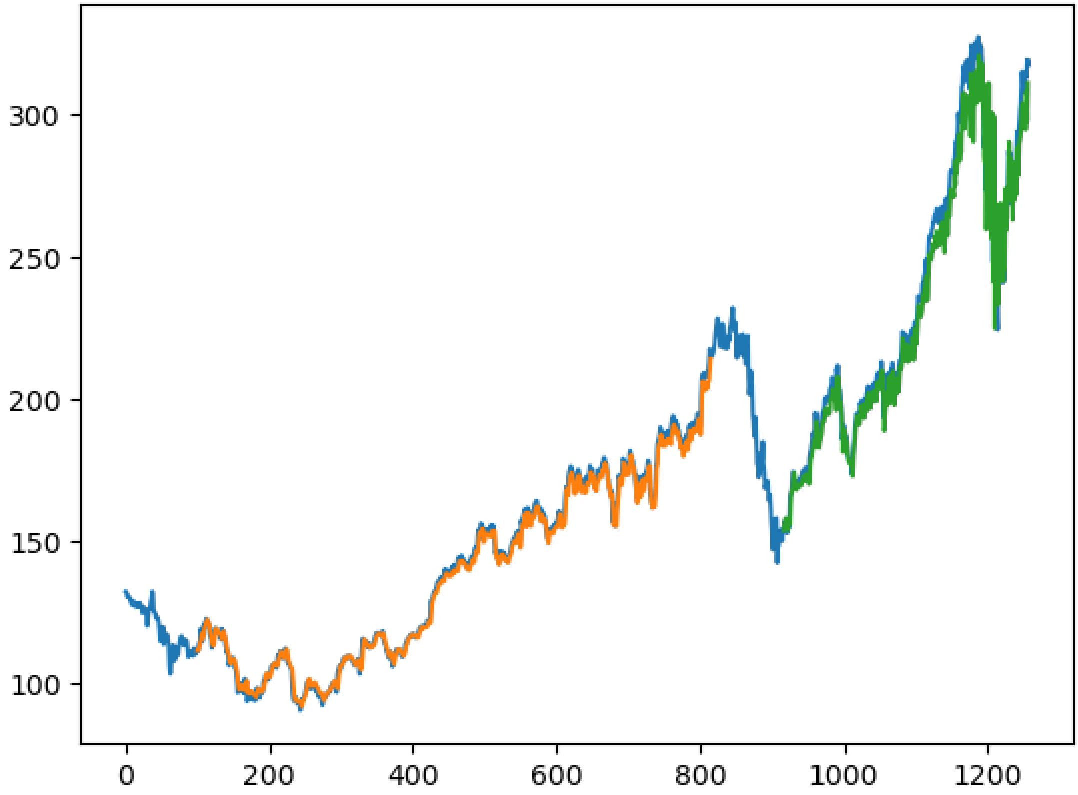
Out[104]:
```
1258
```

Plotting Actual, Training and Testing graph together

In [99]:
```
time_step=100
trainPredictPlot = numpy.empty_like(df1)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[time_step:len(train_predict)+time_step, :] = train_predict

testPredictPlot = numpy.empty_like(df1)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(train_predict)+(time_step*2)+1:len(df1)-1, :] = test_predict

plt.plot(mmscaler.inverse_transform(df1))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```



In the above graph

1). Blue line --> acutal data

2).Orange --> training predicted data

3). Green -->testing predicted data

In [ ]: