In [ ]:
```
import pandas as pd
```

In [ ]:
```
df=pd.read_csv("https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TATAGLOBAL.c
```

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

Out[ ]:

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
1	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
3	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
4	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55

In [ ]:
```
#Close price forecasting
data=df["Close"]
data
```

Out[ ]:
```
0       233.75
1       233.25
2       234.25
3       236.10
4       233.30
         ...
2030    118.65
2031    117.60
2032    120.65
2033    120.90
2034    121.55
Name: Close, Length: 2035, dtype: float64
```

In [ ]:
```
#MinmaxScaler
from sklearn.preprocessing import MinMaxScaler
import numpy as np
minmax=MinMaxScaler(feature_range=(0,1))
df1=minmax.fit_transform(np.array(data).reshape(-1,1))
```

In [ ]:
```
df1.shape , df
```

Out[ ]:
```
((2035, 1),
             Date    Open    High     Low    Last   Close  \
0      2018-09-28  234.05  235.95  230.20  233.50  233.75
1      2018-09-27  234.55  236.80  231.10  233.80  233.25
2      2018-09-26  240.00  240.00  232.50  235.00  234.25
3      2018-09-25  233.30  236.75  232.00  236.25  236.10
4      2018-09-24  233.55  239.20  230.75  234.00  233.30
...           ...     ...     ...     ...     ...     ...
2030   2010-07-27  117.60  119.50  112.00  118.80  118.65
2031   2010-07-26  120.10  121.00  117.10  117.10  117.60
2032   2010-07-23  121.80  121.95  120.25  120.35  120.65
2033   2010-07-22  120.30  122.00  120.25  120.75  120.90
2034   2010-07-21  122.10  123.00  121.05  121.10  121.55

      Total Trade Quantity  Turnover (Lacs)
0                  3069914          7162.35
1                  5082859         11859.95
2                  2240909          5248.60
3                  2349368          5503.90
4                  3423509          7999.55
...                    ...              ...
2030                586100           694.98
2031                658440           780.01
2032                281312           340.31
2033                293312           355.17
2034                658666           803.56

[2035 rows x 8 columns])
```

In [ ]:
```
#Train-Test Split
train_size=round(len(df)*.75)
train_size
```

Out[ ]:
```
1526
```

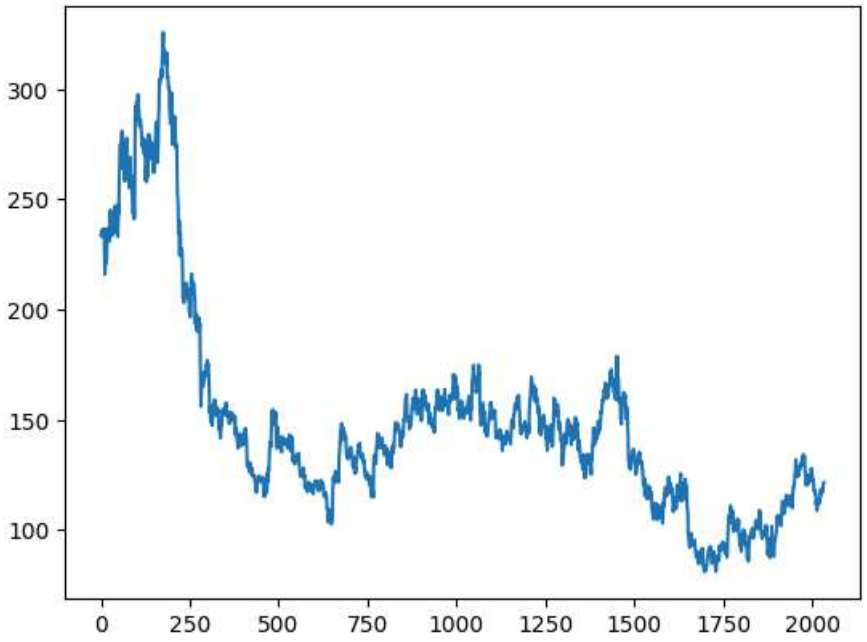
In [ ]:
```
train=df1[:train_size]
train,train.shape
```

Out[ ]:
```
(array([[0.62418301],
        [0.62214052],
        [0.62622549],
        ...,
        [0.18831699],
        [0.18811275],
        [0.17034314]]),
 (1526, 1))
```

In [ ]:
```
test=df1[train_size:]
test,test.shape
```

In [ ]:
```
import matplotlib.pyplot as plt
plt.plot(data)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7895e39dbf40>]
```



In [ ]:
```
#data Preprocessing to train and test LSTM MODEL
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
    dataX.append(a)
    dataY.append(dataset[i + time_step, 0])
  return numpy.array(dataX), numpy.array(dataY)
```

In [ ]:
```
X_train,Y_train=create_dataset(train,100)
X_test,Y_test=create_dataset(test,100)
```

In [ ]:
```
X_train.shape,Y_train.shape
```

Out[ ]:
```
((1425, 100), (1425,))
```

In [ ]:
```
X_test.shape,Y_test.shape
```

Out[ ]:
```
((408, 100), (408,))
```

In [ ]:
```
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)
X_test.shape
```

Out[ ]:
```
(408, 100, 1)
```

In [ ]:
```
#LSTM Model creating
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense,Bidirectional
```

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

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

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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 bidirectional (Bidirection  (None, 100, 100)          40400
 al)

 bidirectional_1 (Bidirecti  (None, 100)               60400
 onal)

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 111301 (434.77 KB)
Trainable params: 111301 (434.77 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:
```
#train the Model
hist=model.fit(X_train,Y_train, validation_data=(X_test,Y_test),epochs=20,batch_size=64)
```

```
Epoch 1/20
23/23 [==============================] - 12s 132ms/step - loss: 0.0939 - val_loss: 0.0438
Epoch 2/20
23/23 [==============================] - 1s 27ms/step - loss: 0.0412 - val_loss: 0.0291
Epoch 3/20
23/23 [==============================] - 1s 23ms/step - loss: 0.0275 - val_loss: 0.0382
Epoch 4/20
23/23 [==============================] - 1s 24ms/step - loss: 0.0269 - val_loss: 0.0241
Epoch 5/20
23/23 [==============================] - 1s 23ms/step - loss: 0.0254 - val_loss: 0.0181
Epoch 6/20
23/23 [==============================] - 1s 24ms/step - loss: 0.0224 - val_loss: 0.0177
Epoch 7/20
23/23 [==============================] - 1s 24ms/step - loss: 0.0208 - val_loss: 0.0178
Epoch 8/20
23/23 [==============================] - 1s 25ms/step - loss: 0.0213 - val_loss: 0.0154
Epoch 9/20
23/23 [==============================] - 1s 23ms/step - loss: 0.0205 - val_loss: 0.0159
Epoch 10/20
23/23 [==============================] - 1s 24ms/step - loss: 0.0195 - val_loss: 0.0153
Epoch 11/20
23/23 [==============================] - 1s 24ms/step - loss: 0.0196 - val_loss: 0.0148
Epoch 12/20
23/23 [==============================] - 1s 24ms/step - loss: 0.0186 - val_loss: 0.0161
Epoch 13/20
23/23 [==============================] - 1s 25ms/step - loss: 0.0174 - val_loss: 0.0151
Epoch 14/20
23/23 [==============================] - 1s 30ms/step - loss: 0.0185 - val_loss: 0.0156
Epoch 15/20
23/23 [==============================] - 1s 24ms/step - loss: 0.0167 - val_loss: 0.0136
Epoch 16/20
23/23 [==============================] - 1s 23ms/step - loss: 0.0170 - val_loss: 0.0135
Epoch 17/20
23/23 [==============================] - 1s 28ms/step - loss: 0.0167 - val_loss: 0.0131
Epoch 18/20
23/23 [==============================] - 1s 33ms/step - loss: 0.0167 - val_loss: 0.0131
Epoch 19/20
23/23 [==============================] - 1s 32ms/step - loss: 0.0167 - val_loss: 0.0132
Epoch 20/20
23/23 [==============================] - 1s 28ms/step - loss: 0.0184 - val_loss: 0.0123
```

In [ ]:
```
#predicting model
X_train_pred=model.predict(X_train)
X_test_pred=model.predict(X_test)
```

```
45/45 [==============================] - 2s 10ms/step
13/13 [==============================] - 0s 9ms/step
```

In [ ]:
```
train_predict=minmax.inverse_transform(X_train_pred)
test_predict=minmax.inverse_transform(X_test_pred)
```

In [ ]:
```
Y_test=Y_test.reshape(np.array(Y_test).shape[0],1)
```

In [ ]:
```
Y_test=minmax.inverse_transform(Y_test)
Y_test
```

In [ ]:
```
from sklearn.metrics import mean_absolute_error
import math
math.sqrt(mean_absolute_error(Y_test,test_predict))
```

Out[ ]:
```
1.7360784416366168
```

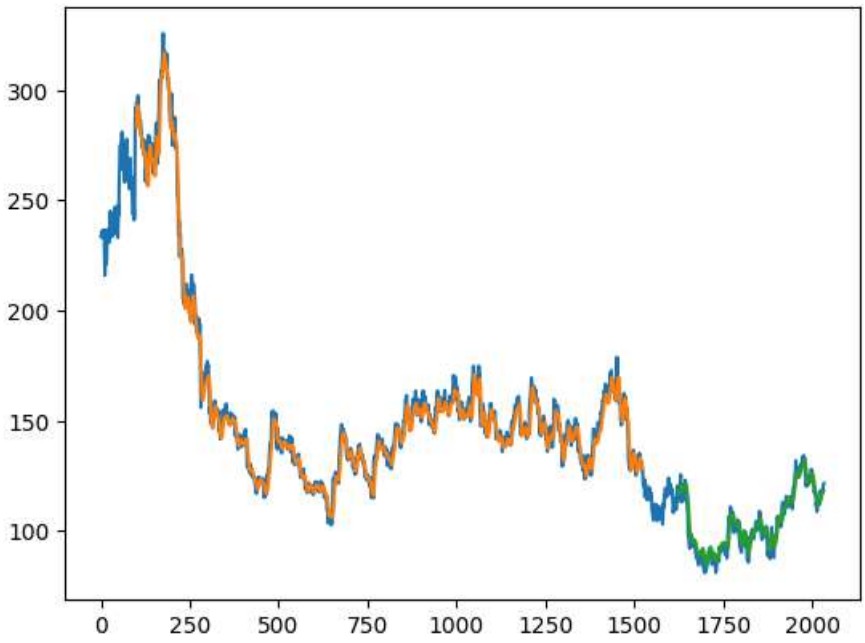
In [ ]:
```
import math
error=abs(Y_test-test_predict).mean()
error
```

Out[ ]:
```
3.013968355515424
```

In [ ]:
```
len(Y_test)
```

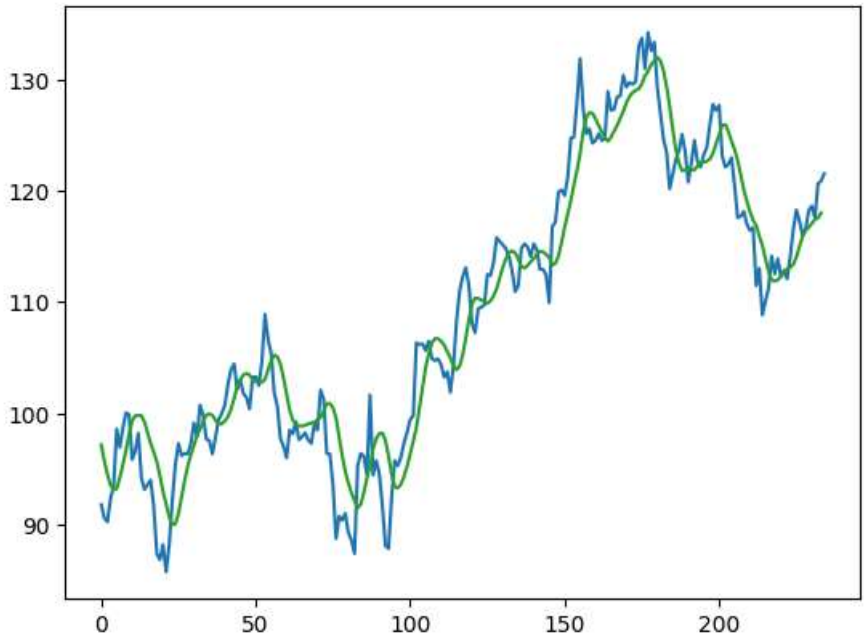
Out[ ]:
```
408
```

In [ ]:
```
# shift train predictions for plotting
look_back=100
trainPredictPlot = numpy.empty_like(df1)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, : ] = train_predict
# shift test predictions for plotting
testPredictPlot = numpy.empty_like(df1)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(df1)-1, :] = test_predict
# plot baseline and predictions
v=0
plt.plot(minmax.inverse_transform(df1)[v:])
plt.plot(trainPredictPlot[v:])
plt.plot(testPredictPlot[v:])
plt.show()
```



In [ ]:
```
# shift train predictions for plotting
look_back=100
trainPredictPlot = numpy.empty_like(df1)
```

```
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, : ] = train_predict
# shift test predictions for plotting
testPredictPlot = numpy.empty_like(df1)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(df1)-1, :] = test_predict
# plot baseline and predictions
v=1800
plt.plot(minmax.inverse_transform(df1)[v:])
plt.plot(trainPredictPlot[v:])
plt.plot(testPredictPlot[v:])
plt.show()
```

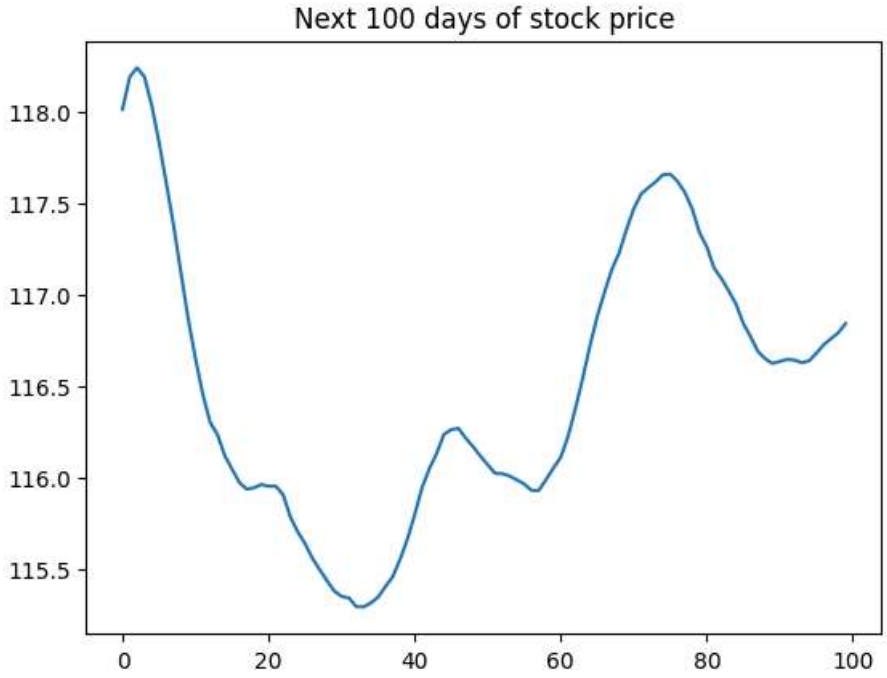


In [ ]:
```
#forecasting nest 100 days data
list1=(X_test[-1]).reshape(100).tolist()
list1
final_list=[]

for i in range(0,100):
  new_value=model.predict(np.array(list1).reshape(1,100,1))
  list1.append(new_value[0][0])
  list1.pop(0)
  minmax_vale=minmax.inverse_transform(new_value)[0][0]
  final_list.append(minmax_vale)
```

In [ ]:
```
#Next 100 days of stock price"
plt.title("Next 100 days of stock price")
plt.plot(final_list)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x78950d0e6a10>]
```


Next 100 days of stock price
118.0
117.5
117.0
116.5
116.0
115.5
0
20
40
60
80
100