# STOCK MARKET PREDICTION BY USING LSTM

IMPORTING LIBRARIES

In [ ]:
```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
```

In [2]:
```
# uploading the data
df= pd.read_csv('C:/Users/Harshita/OneDrive/Desktop/Microsoft Stocks.csv')
```

In [3]:
```
df
```

Out[3]:

| | Date | Price | Open | High | Low | Volume |
|---|---|---|---|---|---|---|
| **0** | 05/15/2023 | 309.46 | 309.10 | 309.90 | 307.59 | 16290000 |
| **1** | 05/12/2023 | 308.97 | 310.55 | 310.65 | 306.60 | 19770000 |
| **2** | 05/11/2023 | 310.11 | 310.10 | 311.12 | 306.26 | 31680000 |
| **3** | 05/10/2023 | 312.31 | 308.62 | 313.00 | 307.67 | 30080000 |
| **4** | 05/09/2023 | 307.00 | 308.00 | 310.04 | 306.31 | 21340000 |
| **...** | ... | ... | ... | ... | ... | ... |
| **9364** | 03/20/1986 | 0.10 | 0.10 | 0.10 | 0.09 | 58440000 |
| **9365** | 03/19/1986 | 0.10 | 0.10 | 0.10 | 0.10 | 47890000 |
| **9366** | 03/18/1986 | 0.10 | 0.10 | 0.10 | 0.10 | 66470000 |
| **9367** | 03/17/1986 | 0.10 | 0.10 | 0.10 | 0.10 | 133169999 |
| **9368** | 03/14/1986 | 0.10 | 0.10 | 0.10 | 0.10 | 308160000 |

9369 rows × 6 columns

In [4]:
```
df.head()
```

Out[4]:

| | Date | Price | Open | High | Low | Volume |
|---|---|---|---|---|---|---|
| **0** | 05/15/2023 | 309.46 | 309.10 | 309.90 | 307.59 | 16290000 |
| **1** | 05/12/2023 | 308.97 | 310.55 | 310.65 | 306.60 | 19770000 |
| **2** | 05/11/2023 | 310.11 | 310.10 | 311.12 | 306.26 | 31680000 |
| **3** | 05/10/2023 | 312.31 | 308.62 | 313.00 | 307.67 | 30080000 |
| **4** | 05/09/2023 | 307.00 | 308.00 | 310.04 | 306.31 | 21340000 |

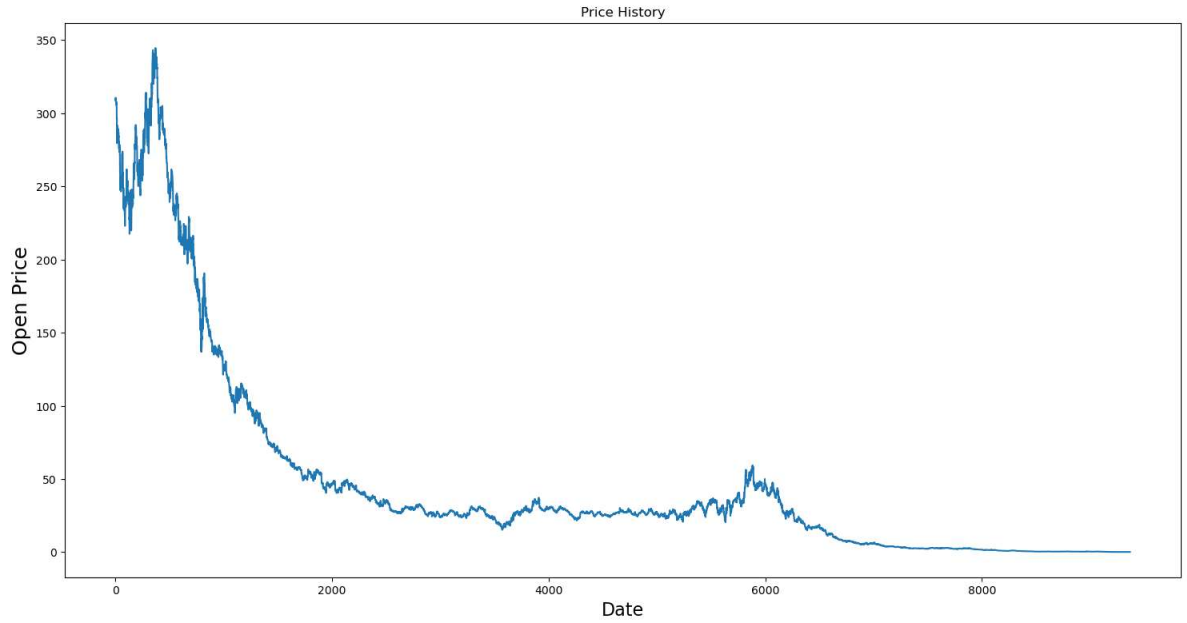
In [5]:
```
df.shape
```

Out[5]:
```
(9369, 6)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9369 entries, 0 to 9368
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    9369 non-null   object 
 1   Price   9369 non-null   float64
 2   Open    9369 non-null   float64
 3   High    9369 non-null   float64
 4   Low     9369 non-null   float64
 5   Volume  9369 non-null   int64  
dtypes: float64(4), int64(1), object(1)
memory usage: 439.3+ KB
```

In [7]:
```
# Ploting the opening price of the stock to visulize the trend
plt.figure(figsize=(18,9))
plt.title('Price History')
plt.plot(df['Open'])
plt.xlabel('Date',fontsize=16)
plt.ylabel('Open Price',fontsize=18)
plt.show()
```




# preprocess the data before feeding it into the LSTM.

In [8]:
```
df1=df.reset_index()['Open']
```

In [9]:
```
df1
```

Out[9]:
```
0       309.10
1       310.55
2       310.10
3       308.62
4       308.00
         ...
9364      0.10
9365      0.10
9366      0.10
9367      0.10
9368      0.10
Name: Open, Length: 9369, dtype: float64
```

Normalize the data between 0 and 1 using the MinMaxScaler:

In [10]:
```
scaler=MinMaxScaler(feature_range=(0,1))
df1=scaler.fit_transform(np.array(df1).reshape(-1,1))
```

# Train the Dataset

In [11]:
```
train_data = df1[:int(len(df1)*0.8)]
x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

In [12]:
```
x_train,y_train
```

Out[12]:
```
(array([[[0.89690303],
         [0.90111166],
         [0.89980553],
         ...,
         [0.73723623],
         [0.73836821],
         [0.75261951]],

        [[0.90111166],
         [0.89980553],
         [0.89550982],
         ...,
         [0.73836821],
         [0.75261951],
         [0.76605811]],

        [[0.89980553],
         [0.89550982],
         [0.89371027],
         ...,
         [0.75261951],
         [0.76605811],
         [0.77853888]],

        ...,

        [[0.00690796],
         [0.00664674],
         [0.00687894],
         ...,
         [0.00679186],
         [0.00682089],
         [0.00661771]],

        [[0.00664674],
         [0.00687894],
         [0.00699504],
         ...,
         [0.00682089],
         [0.00661771],
         [0.00644356]],

        [[0.00687894],
         [0.00699504],
         [0.00711114],
         ...,
         [0.00661771],
         [0.00644356],
         [0.00632746]]]),
 array([0.76605811, 0.77853888, 0.79116478, ..., 0.00644356, 0.00632746,
        0.00621136]))
```

In [13]:
```
x_train.shape
```

Out[13]:
```
(7435, 60, 1)
```

In [14]:
```
y_train.shape
```

Out[14]:
```
(7435,)
```

# Build the LSTM Model

In [15]:
```
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(50, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(50))
model.add(Dropout(0.2))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(x_train, y_train, epochs=50, batch_size=32)
```

```
Epoch 1/50
233/233 [==============================] - 22s 60ms/step - loss: 0.0021
Epoch 2/50
233/233 [==============================] - 14s 60ms/step - loss: 8.3543e-04
Epoch 3/50
233/233 [==============================] - 14s 60ms/step - loss: 8.0258e-04
Epoch 4/50
233/233 [==============================] - 14s 59ms/step - loss: 8.0756e-04
Epoch 5/50
233/233 [==============================] - 14s 58ms/step - loss: 6.7440e-04
Epoch 6/50
233/233 [==============================] - 14s 59ms/step - loss: 6.3934e-04
Epoch 7/50
233/233 [==============================] - 14s 58ms/step - loss: 5.6438e-04
Epoch 8/50
233/233 [==============================] - 14s 58ms/step - loss: 5.8888e-04
Epoch 9/50
233/233 [==============================] - 14s 59ms/step - loss: 5.3729e-04
Epoch 10/50
233/233 [==============================] - 14s 58ms/step - loss: 4.9688e-04
Epoch 11/50
233/233 [==============================] - 14s 59ms/step - loss: 5.1526e-04
Epoch 12/50
233/233 [==============================] - 14s 59ms/step - loss: 5.6324e-04
Epoch 13/50
233/233 [==============================] - 14s 59ms/step - loss: 5.1884e-04
Epoch 14/50
233/233 [==============================] - 14s 59ms/step - loss: 5.0713e-04
Epoch 15/50
233/233 [==============================] - 14s 58ms/step - loss: 4.4020e-04
Epoch 16/50
233/233 [==============================] - 14s 58ms/step - loss: 4.3688e-04
Epoch 17/50
233/233 [==============================] - 14s 59ms/step - loss: 4.4310e-04
Epoch 18/50
233/233 [==============================] - 14s 59ms/step - loss: 4.8443e-04
Epoch 19/50
233/233 [==============================] - 14s 59ms/step - loss: 4.2100e-04
Epoch 20/50
233/233 [==============================] - 14s 58ms/step - loss: 5.0525e-04
Epoch 21/50
233/233 [==============================] - 14s 58ms/step - loss: 4.5365e-04
Epoch 22/50
233/233 [==============================] - 14s 59ms/step - loss: 4.7283e-04
Epoch 23/50
233/233 [==============================] - 14s 58ms/step - loss: 4.6749e-04
Epoch 24/50
233/233 [==============================] - 14s 59ms/step - loss: 4.9754e-04
Epoch 25/50
233/233 [==============================] - 14s 59ms/step - loss: 4.5034e-04
Epoch 26/50
233/233 [==============================] - 14s 58ms/step - loss: 4.3131e-04
Epoch 27/50
233/233 [==============================] - 14s 59ms/step - loss: 4.2827e-04
Epoch 28/50
233/233 [==============================] - 14s 59ms/step - loss: 4.7306e-04
Epoch 29/50
```

```
233/233 [==============================] - 14s 59ms/step - loss: 4.3821e-04
Epoch 30/50
233/233 [==============================] - 14s 59ms/step - loss: 4.3309e-04
Epoch 31/50
233/233 [==============================] - 14s 59ms/step - loss: 4.0134e-04
Epoch 32/50
233/233 [==============================] - 14s 59ms/step - loss: 4.1012e-04
Epoch 33/50
233/233 [==============================] - 14s 59ms/step - loss: 4.0519e-04
Epoch 34/50
233/233 [==============================] - 14s 59ms/step - loss: 4.1713e-04
Epoch 35/50
233/233 [==============================] - 14s 59ms/step - loss: 4.5183e-04
Epoch 36/50
233/233 [==============================] - 14s 58ms/step - loss: 4.6539e-04
Epoch 37/50
233/233 [==============================] - 14s 58ms/step - loss: 4.6070e-04
Epoch 38/50
233/233 [==============================] - 14s 58ms/step - loss: 4.3267e-04
Epoch 39/50
233/233 [==============================] - 14s 58ms/step - loss: 4.2955e-04
Epoch 40/50
233/233 [==============================] - 14s 59ms/step - loss: 4.3790e-04
Epoch 41/50
233/233 [==============================] - 14s 59ms/step - loss: 4.6440e-04
Epoch 42/50
233/233 [==============================] - 14s 59ms/step - loss: 4.3452e-04
Epoch 43/50
233/233 [==============================] - 14s 59ms/step - loss: 4.2909e-04
Epoch 44/50
233/233 [==============================] - 14s 59ms/step - loss: 4.0784e-04
Epoch 45/50
233/233 [==============================] - 14s 59ms/step - loss: 3.9006e-04
Epoch 46/50
233/233 [==============================] - 14s 59ms/step - loss: 4.4194e-04
Epoch 47/50
233/233 [==============================] - 14s 59ms/step - loss: 4.4934e-04
Epoch 48/50
233/233 [==============================] - 14s 59ms/step - loss: 3.8053e-04
Epoch 49/50
233/233 [==============================] - 14s 58ms/step - loss: 4.0182e-04
Epoch 50/50
233/233 [==============================] - 19s 81ms/step - loss: 3.8539e-04
```

Out[15]: `<keras.src.callbacks.History at 0x1634cb7b1d0>`

# Testing the Dataset

In [44]:
```
test_data = df1[int(len(df1)*0.8) - 60:]
x_test = []
y_test = df[int(len(df)*0.8):]


for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])

x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

```
59/59 [==============================] - 2s 38ms/step
```

In [45]:
```
x_test.shape
```

Out[45]:
```
(1874, 60, 1)
```

In [46]:
```
train_predict=model.predict(x_train)
```

```
233/233 [==============================] - 8s 36ms/step
```

In [47]:
```
test_predict=model.predict(x_test)
```

```
59/59 [==============================] - 2s 35ms/step
```

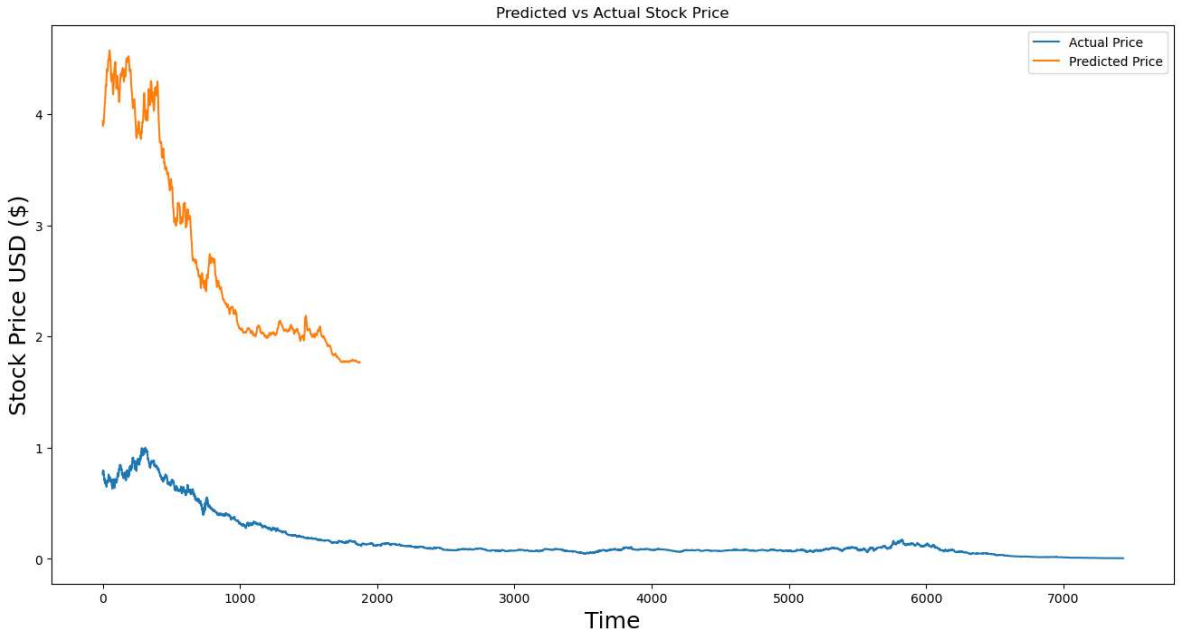
In [48]:
```
train_predict=scaler.inverse_transform(train_predict)
test_predict=scaler.inverse_transform(test_predict)
```

In [49]:
```
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,train_predict))
```

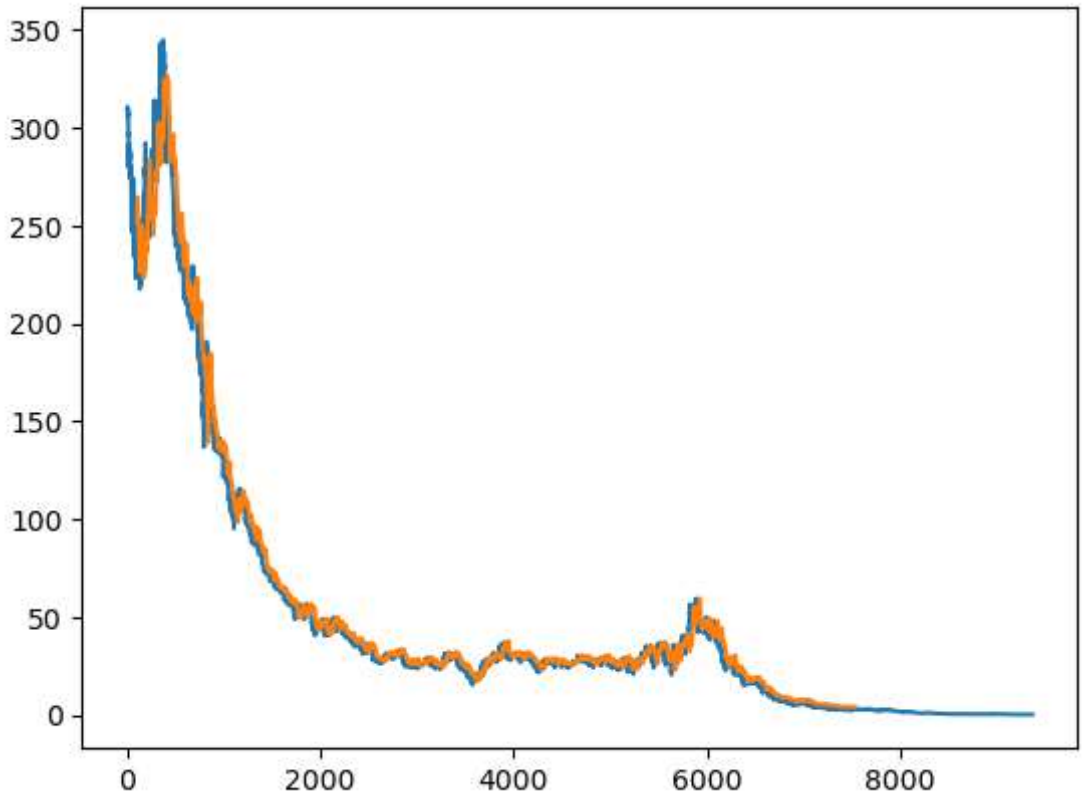
Out[49]:
```
91.08515833548249
```

# Visualizing the Predicted price as compared to Actual price

In [50]:
```
plt.figure(figsize=(16,8))
plt.title('Predicted vs Actual Stock Price')
plt.plot(y_train, label='Actual Price')
plt.plot(predictions, label='Predicted Price')
plt.xlabel('Time', fontsize=18)
plt.ylabel('Stock Price USD ($)', fontsize=18)
plt.legend()
plt.show()
```



In [61]:
```
# shift train predictions for plotting
trainPredictPlot = np.empty_like(df1)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict
# plot baseline and predictions
plt.plot(scaler.inverse_transform(df1))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```




In [62]:
```
len(test_data)
```

Out[62]: 1934

In [63]:
```
x_input=test_data[1834:].reshape(1,-1)
x_input.shape
```

Out[63]: (1, 100)

In [64]:
```
temp_input=list(x_input)
temp_input=temp_input[0].tolist()
```

In [65]:
```
temp_input
```

Out[65]:
```
[2.902504861695645e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 2.902504861695645e-05,
 5.8050097233912845e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 0.0,
 2.902504861695645e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 2.902504861695645e-05,
 2.902504861695645e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 8.707514585086924e-05,
 8.707514585086924e-05,
 8.707514585086924e-05,
 8.707514585086924e-05,
 8.707514585086924e-05,
 8.707514585086924e-05,
 8.707514585086924e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
 5.8050097233912845e-05,
```

```
5.8050097233912845e-05,
5.8050097233912845e-05,
5.8050097233912845e-05,
5.8050097233912845e-05,
5.8050097233912845e-05,
5.8050097233912845e-05,
5.8050097233912845e-05,
5.8050097233912845e-05,
5.8050097233912845e-05,
5.8050097233912845e-05,
5.8050097233912845e-05,
5.8050097233912845e-05,
8.707514585086924e-05,
8.707514585086924e-05,
5.8050097233912845e-05,
2.902504861695645e-05,
2.902504861695645e-05,
2.902504861695645e-05,
2.902504861695645e-05,
5.8050097233912845e-05,
2.902504861695645e-05,
2.902504861695645e-05,
2.902504861695645e-05,
2.902504861695645e-05,
2.902504861695645e-05,
2.902504861695645e-05,
2.902504861695645e-05,
0.0,
2.902504861695645e-05,
2.902504861695645e-05,
2.902504861695645e-05,
0.0,
2.902504861695645e-05,
2.902504861695645e-05,
0.0,
0.0,
0.0,
0.0,
2.902504861695645e-05,
2.902504861695645e-05,
2.902504861695645e-05,
2.902504861695645e-05,
2.902504861695645e-05]
```

# Prediction for next 10 days

In [86]:
```
from numpy import array

lst_output=[]
n_steps=100
i=0
while(i<30):

    if(len(temp_input)>100):
        #print(temp_input)
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1
```

```
[0.00487864]
101
1 day input [2.90250486e-05 2.90250486e-05 2.90250486e-05 2.90250486e-05
 2.90250486e-05 5.80500972e-05 5.80500972e-05 5.80500972e-05
 2.90250486e-05 5.80500972e-05 2.90250486e-05 2.90250486e-05
 2.90250486e-05 2.90250486e-05 2.90250486e-05 2.90250486e-05
 2.90250486e-05 0.00000000e+00 2.90250486e-05 2.90250486e-05
 2.90250486e-05 5.80500972e-05 5.80500972e-05 5.80500972e-05
 5.80500972e-05 2.90250486e-05 2.90250486e-05 5.80500972e-05
 5.80500972e-05 5.80500972e-05 5.80500972e-05 5.80500972e-05
 5.80500972e-05 5.80500972e-05 5.80500972e-05 5.80500972e-05
 5.80500972e-05 5.80500972e-05 5.80500972e-05 5.80500972e-05
 8.70751459e-05 8.70751459e-05 8.70751459e-05 8.70751459e-05
 8.70751459e-05 8.70751459e-05 8.70751459e-05 5.80500972e-05
 5.80500972e-05 5.80500972e-05 5.80500972e-05 5.80500972e-05
 5.80500972e-05 5.80500972e-05 5.80500972e-05 5.80500972e-05
 5.80500972e-05 5.80500972e-05 5.80500972e-05 5.80500972e-05
 5.80500972e-05 5.80500972e-05 5.80500972e-05 5.80500972e-05
 5.80500972e-05 5.80500972e-05 5.80500972e-05 5.80500972e-05
```

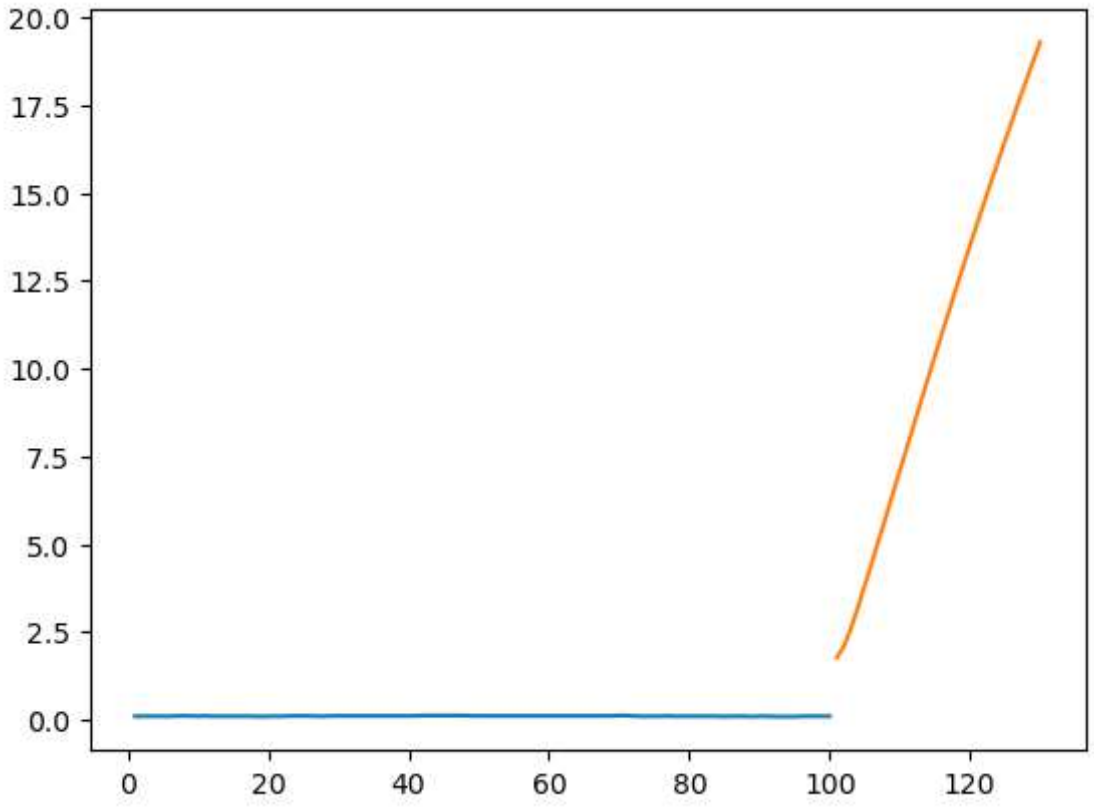
In [88]:
```
day_new=np.arange(1,101)
day_pred=np.arange(101,131)
```

In [89]:
```
len(df1)
```

Out[89]:
```
9369
```

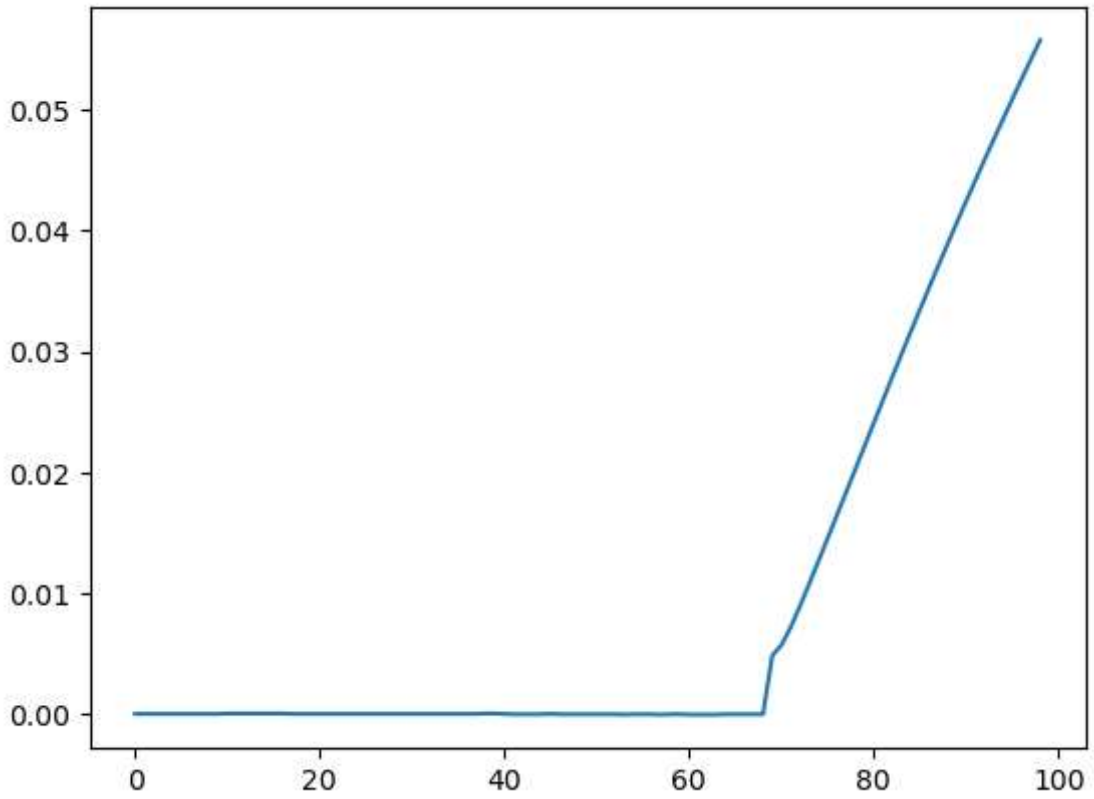
In [97]:
```
plt.plot(day_new,scaler.inverse_transform(df1[9269:]))
plt.plot(day_pred,scaler.inverse_transform(lst_output))
```

Out[97]:
```
[<matplotlib.lines.Line2D at 0x1636304f890>]
```



In [99]:
```
df3=df1.tolist()
df3.extend(lst_output)
plt.plot(df3[9300:])
```

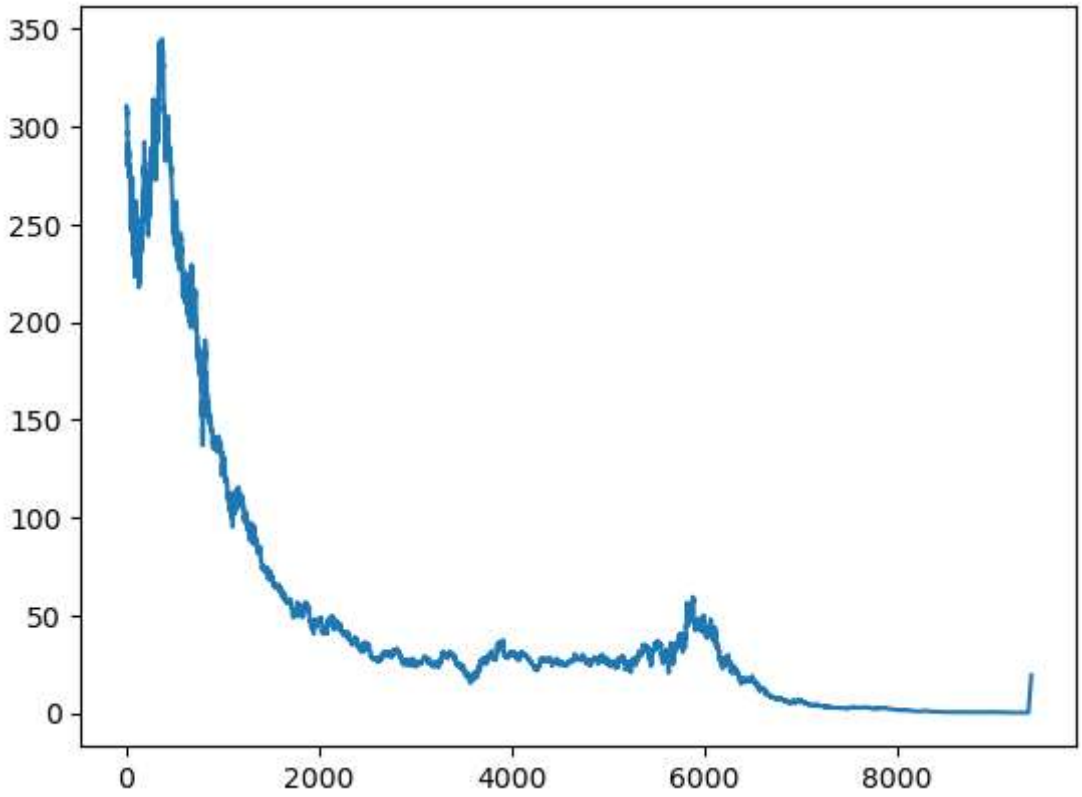
Out[99]:
```
[<matplotlib.lines.Line2D at 0x163626c40d0>]
```




In [100]:
```
df3=scaler.inverse_transform(df3).tolist()
```

In [101]:
```
plt.plot(df3)
```

Out[101]:
```
[<matplotlib.lines.Line2D at 0x163627e4650>]
```




# THANK YOU