```
#!/usr/bin/env python
# coding: utf-8

# In[9]:
# Installing yfinance package using pip
pip install yfinance

# Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf

# In[12]:
# Defining the start and end dates for fetching stock data, and the stock symbol
start = '2010-01-01'
end = '2025-02-17'
stock = 'GOOG'

# Fetching historical stock market data using yfinance
data = yf.download(stock, start, end)

# In[13]:
# Resetting the index of the DataFrame
data.reset_index(inplace=True)

# In[15]:
# Calculating the 100-day moving average
ma_100_days = data.Close.rolling(100).mean()

# Plotting the 100-day moving average and closing price
plt.figure(figsize=(8, 6))
plt.plot(ma_100_days, color="red", label='MA 100 Days')
plt.plot(data.Close, color="green", label="Close")
plt.legend()
plt.show()

# In[17]:
# Calculating the 200-day moving average
ma_200_days = data.Close.rolling(200).mean()

# Plotting the 100-day and 200-day moving averages along with closing price
plt.figure(figsize=(8, 6))
plt.plot(ma_100_days, color="red", label='MA 100 Days')
plt.plot(ma_200_days, color="blue", label='MA 200 Days')
plt.plot(data.Close, color="green", label="Close")
plt.legend()
plt.show()
```

```
# In[19]:
# Dropping any null or missing values from the DataFrame
data.dropna(inplace=True)

# In[20]:
# Splitting the data into training and testing sets (80% training, 20% testing)
data_train = pd.DataFrame(data.Close[0:int(len(data)*0.80)])
data_test = pd.DataFrame(data.Close[int(len(data)*0.80):len(data)])

# In[22]:
# Scaling the training data between 0 and 1 using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
data_train_scale = scaler.fit_transform(data_train)

# In[24]:
# Preparing the training data by creating sequences of 100 data points for input and the next
data point for output
x = []
y = []

for i in range(100, data_train_scale.shape[0]):
    x.append(data_train_scale[i - 100: i])
    y.append(data_train_scale[i, 0])

x, y = np.array(x), np.array(y)

# In[27]:
# Defining the LSTM model architecture using Keras Sequential API
from keras.layers import Dense, Dropout, LSTM
from keras.models import Sequential

model = Sequential()
model.add(LSTM(units=50, activation='relu', return_sequences=True,
input_shape=(x.shape[1], 1)))
model.add(Dropout(0.2))

model.add(LSTM(units=60, activation='relu', return_sequences=True))
model.add(Dropout(0.3))

model.add(LSTM(units=80, activation='relu', return_sequences=True))
model.add(Dropout(0.4))

model.add(LSTM(units=120, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(units=1))
```

```
# In[28]:
# Compiling the model with Adam optimizer and mean squared error loss function
model.compile(optimizer='adam', loss='mean_squared_error')

# In[29]:
# Training the model on the training data
model.fit(x, y, epochs=50, batch_size=32, verbose=1)

# In[31]:
# Concatenating the last 100 days of training data with the testing data
pas_100_days = data_train.tail(100)
data_test = pd.concat([pas_100_days, data_test], ignore_index=True)

# In[36]:
# Scaling the testing data
data_test_scale = scaler.fit_transform(data_test)

# In[37]:
# Preparing the testing data by creating sequences of 100 data points for input and the next
data point for output
x = []
y = []

for i in range(100, data_test_scale.shape[0]):
    x.append(data_test_scale[i - 100: i])
    y.append(data_test_scale[i, 0])

x, y = np.array(x), np.array(y)

# In[39]:
# Predicting the values using the trained model
y_predict = model.predict(x)

# In[42]:
# Rescaling the predicted and actual values to the original scale
scale = 1/scaler.scale_
y_predict = y_predict*scale
y = y*scale

# In[43]:
# Plotting the predicted and actual values
plt.figure(figsize=(10, 8))
plt.plot(y_predict, color='red', label='Predicted Price')
plt.plot(y, color='green', label='Original Price')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()

# In[60]:
# Preparing input data for predicting future stock prices
x_input = data_test.iloc[711:].values.reshape(1, -1)
x_input.shape

# In[61]:
# Converting input data to list
temp_input = list(x_input)
temp_input = temp_input[0].tolist()

# In[62]:
# Generating future stock price predictions for the next 30 days
lst_output=[]
n_steps=100
i=0
while(i<30):

    if(len(temp_input)>100):
        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))
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        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

print(lst_output)

# In[68]:
# Plotting original and predicted values for the next 30 days
day_new=np.arange(1,101)
day_pred=np.arange(101,131)

plt.plot(day_new,scaler.inverse_transform(data_train[2744:]))
plt.plot(day_pred,scaler.inverse_transform(lst_output))
```

```
# In[44]:
# Calculating Mean Absolute Error (MAE)
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y, y_predict)
print("Mean Absolute Error (MAE) :", mae)

# In[46]:
# Saving the trained model
model.save("Stock Prediction Model.keras")
```