```
import yfinance as yf
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 LSTM, Dense

# 1. Download stock data
def get_stock_data(ticker, start, end):
    data = yf.download(ticker, start=start, end=end)
    return data[['Close']], data

# 2. Preprocess the data
def preprocess_data(data, time_step=60):
    scaler = MinMaxScaler(feature_range=(0, 1))
    data_scaled = scaler.fit_transform(data)

    X, y = [], []
    for i in range(time_step, len(data_scaled)):
        X.append(data_scaled[i - time_step:i])
        y.append(data_scaled[i])

    X = np.array(X)
    y = np.array(y)

    return X, y, scaler

# 3. Build the LSTM model
def build_lstm_model(input_shape):
    model = Sequential()
    model.add(LSTM(units=50, return_sequences=True, input_shape=input_shape))
    model.add(LSTM(units=50))
    model.add(Dense(units=1))
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

# MAIN BLOCK
if __name__ == "__main__":
    ticker = 'AAPL'  # You can change this to any stock symbol
    start_date = '2015-01-01'
    end_date = '2023-12-31'

    data, raw_df = get_stock_data(ticker, start_date, end_date)
    X, y, scaler = preprocess_data(data)

    # Reshape input to be 3D for LSTM [samples, time steps, features]
    X = X.reshape((X.shape[0], X.shape[1], 1))

    # Split into train and test
    split = int(len(X) * 0.8)
    X_train, X_test = X[:split], X[split:]
    y_train, y_test = y[:split], y[split:]

    # Build and train model
```

```
model = build_lstm_model((X.shape[1], 1))
model.fit(X_train, y_train, epochs=20, batch_size=32, verbose=1)

# Predict
predicted = model.predict(X_test)
predicted_prices = scaler.inverse_transform(predicted)
real_prices = scaler.inverse_transform(y_test.reshape(-1, 1))

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(real_prices, color='blue', label='Actual Price')
plt.plot(predicted_prices, color='red', label='Predicted Price')
plt.title(f'{ticker} Stock Price Prediction')
plt.xlabel('Days')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```