```
#CODE FOR LSTM:
import pandas as pd
import yfinance as yf
import datetime
from datetime import date, timedelta
today = date.today()

d1 = today.strftime("%Y-%m-%d")
end_date = d1
d2 = date.today() - timedelta(days=5000)
d2 = d2.strftime("%Y-%m-%d")
start_date = d2
```

```
data = yf.download('NFLX',
                   start=start_date,
                   end=end_date,
                   progress=False)
data["Date"] = data.index
data = data[["Date", "Open", "High", "Low", "Close", "Adj Close", "Volume"]]
data.reset_index(drop=True, inplace=True)
```

```
d1
```

```
'2024-04-09'
```

```
print(data.head())
```

```
        Date       Open       High        Low      Close  Adj Close    Volume
0 2010-08-02  14.860000  15.000000  14.424286  14.554286  14.554286  24295600
1 2010-08-03  14.437143  14.961429  14.104286  14.915714  14.915714  30139200
2 2010-08-04  14.995714  15.500000  14.957143  15.447143  15.447143  24243100
3 2010-08-05  15.318571  15.814286  15.071429  15.790000  15.790000  24296300
4 2010-08-06  15.527143  16.964287  15.411429  16.902857  16.902857  48693400
```

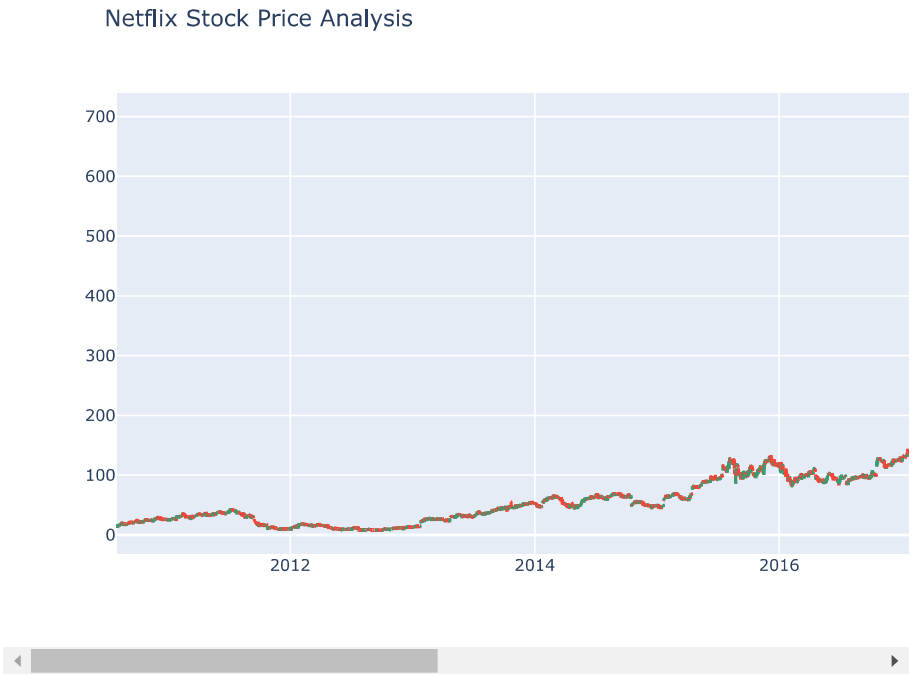
```
print(data.tail())
```

```
           Date        Open        High         Low       Close   Adj Close  \
3439 2024-04-02  611.000000  615.030029  605.510010  614.210022  614.210022
3440 2024-04-03  612.750000  630.409973  611.500000  630.080017  630.080017
3441 2024-04-04  633.210022  638.000000  616.580017  617.140015  617.140015
3442 2024-04-05  624.919983  637.909973  622.710022  636.179993  636.179993
3443 2024-04-08  636.390015  639.000000  628.109985  628.409973  628.409973

       Volume
3439  2029200
3440  2931200
3441  3064300
3442  3372800
3443  2139300
```

```
import plotly.graph_objects as go
figure = go.Figure(data=[go.Candlestick(x=data["Date"],
                                        open=data["Open"],
                                        high=data["High"],
                                        low=data["Low"],
                                        close=data["Close"])])
```

```
figure.update_layout(title = "Netflix Stock Price Analysis",
                     xaxis_rangeslider_visible=False)
figure.show()
```




```
correlation = data.corr()
print(correlation["Close"].sort_values(ascending=False))
x = data[["Open", "High", "Low", "Volume"]]
y = data["Close"]
x = x.to_numpy()
y = y.to_numpy()
y = y.reshape(-1, 1)
```

```
Close        1.000000
Adj Close    1.000000
High         0.999793
Low          0.999782
Open         0.999535
Date         0.869718
Volume      -0.504372
Name: Close, dtype: float64
```

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.2,
                                                random_state=42)
```

```
from keras.models import Sequential
from keras.layers import Dense, LSTM
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (xtrain.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 4, 128)            66560

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 25)                1625

 dense_1 (Dense)             (None, 1)                 26

=================================================================
```

```
Total params: 117619 (459.45 KB)
Trainable params: 117619 (459.45 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
from keras.metrics import MeanAbsolutePercentageError
model.compile(optimizer='adam', loss='mean_squared_error',metrics=[MeanAbsolutePercentageError()])
model.fit(xtrain, ytrain, batch_size=1, epochs=10,validation_split=0.2)
loss, mape = model.evaluate(xtest, ytest, verbose=0)
print("Mean Absolute Percentage Error on test data:", mape)
```

```
Epoch 1/10
2204/2204 [==============================] - 25s 9ms/step - loss: 195.8697 - mean_absolute_percentage_error: 4.2784 - val_loss: 37.7287
Epoch 2/10
2204/2204 [==============================] - 18s 8ms/step - loss: 296.6752 - mean_absolute_percentage_error: 5.0646 - val_loss: 247.5252
Epoch 3/10
2204/2204 [==============================] - 17s 8ms/step - loss: 223.9877 - mean_absolute_percentage_error: 4.2877 - val_loss: 69.6153
Epoch 4/10
2204/2204 [==============================] - 19s 8ms/step - loss: 373.8686 - mean_absolute_percentage_error: 5.3610 - val_loss: 55.0363
Epoch 5/10
2204/2204 [==============================] - 16s 7ms/step - loss: 233.5919 - mean_absolute_percentage_error: 5.8335 - val_loss: 133.4269
Epoch 6/10
2204/2204 [==============================] - 18s 8ms/step - loss: 241.2243 - mean_absolute_percentage_error: 4.8366 - val_loss: 374.4345
Epoch 7/10
2204/2204 [==============================] - 16s 7ms/step - loss: 171.8741 - mean_absolute_percentage_error: 4.4056 - val_loss: 28.1966
Epoch 8/10
2204/2204 [==============================] - 16s 7ms/step - loss: 128.4833 - mean_absolute_percentage_error: 4.1963 - val_loss: 431.7325
Epoch 9/10
2204/2204 [==============================] - 17s 8ms/step - loss: 169.1779 - mean_absolute_percentage_error: 4.5417 - val_loss: 312.7065
Epoch 10/10
2204/2204 [==============================] - 16s 7ms/step - loss: 268.4582 - mean_absolute_percentage_error: 5.4162 - val_loss: 168.1506
Mean Absolute Percentage Error on test data: 3.132878303527832
```

```
import numpy as np
#features = [Open, High, Low, Adj Close, Volume]
features = np.array([[401.970001, 427.700012, 398.200012, 20047500]])
model.predict(features)
```

```
1/1 [==============================] - 0s 46ms/step
array([[428.82822]], dtype=float32)
```