In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('NFLX.csv')
df.head()
```

Out[2]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [4]:
```
df.shape
```

Out[4]:
```
(1009, 7)
```

In [5]:
```
df=df["Close"]
```

In [6]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
```

In [7]:
```
scaler=MinMaxScaler((0,1))
data=scaler.fit_transform(np.array(df).reshape([df.shape[0],1]))
```

In [8]:
```
time_step=100
def createData(data):
    x=[]
    y=[]
    for i in range(len(data)-time_step-1):
        x.append(data[i:(i+time_step)])
        y.append(data[i+time_step])
    return x,y
```

In [9]:
```
x,y=createData(data)
```

In [10]:
```
x=np.array(x)
x=x.reshape(x.shape[0],x.shape[1],1)
y=np.array(y)
```

In [11]:
```
df.shape
```

Out[11]:
```
(1009,)
```

In [12]:
```
xtrain,xtest,ytrain,ytest=x[:int(df.shape[0]*0.8)],x[int(df.shape[0]*0.8):],y[:int(df.shape[0]*0.8)],y[int(df.shape[0]*0.8):]
```

In [13]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,LSTM
```

In [14]:
```
model=Sequential([
    LSTM(128,return_sequences=True,input_shape=xtrain[0].shape),
    LSTM(64,return_sequences=True),
    LSTM(32),
    Dense(16,activation="relu"),
    Dense(1)
])
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),loss="mean_squared_error",metrics=[tf.keras.metrics.RootMeanSquaredError()])
```

In [16]:
```
model.fit(xtrain,ytrain,epochs=25)
```

```
Epoch 1/25
26/26 [==============================] - 3s 128ms/step - loss: 5.6267e-04 - root_mean_squared_error: 0.0237
Epoch 2/25
26/26 [==============================] - 3s 126ms/step - loss: 5.3426e-04 - root_mean_squared_error: 0.0231
Epoch 3/25
26/26 [==============================] - 3s 131ms/step - loss: 5.9049e-04 - root_mean_squared_error: 0.0243
Epoch 4/25
26/26 [==============================] - 3s 128ms/step - loss: 5.4372e-04 - root_mean_squared_error: 0.0233
Epoch 5/25
26/26 [==============================] - 3s 128ms/step - loss: 5.4537e-04 - root_mean_squared_error: 0.0234
Epoch 6/25
26/26 [==============================] - 3s 129ms/step - loss: 7.9172e-04 - root_mean_squared_error: 0.0281
Epoch 7/25
26/26 [==============================] - 3s 129ms/step - loss: 5.8540e-04 - root_mean_squared_error: 0.0242
Epoch 8/25
26/26 [==============================] - 3s 131ms/step - loss: 6.6755e-04 - root_mean_squared_error: 0.0258
Epoch 9/25
26/26 [==============================] - 3s 130ms/step - loss: 5.6983e-04 - root_mean_squared_error: 0.0239
Epoch 10/25
26/26 [==============================] - 3s 128ms/step - loss: 5.8806e-04 - root_mean_squared_error: 0.0242
Epoch 11/25
26/26 [==============================] - 3s 129ms/step - loss: 5.6596e-04 - root_mean_squared_error: 0.0238
Epoch 12/25
26/26 [==============================] - 4s 140ms/step - loss: 5.5713e-04 - root_mean_squared_error: 0.0236
Epoch 13/25
26/26 [==============================] - 4s 141ms/step - loss: 5.1782e-04 - root_mean_squared_error: 0.0228
Epoch 14/25
26/26 [==============================] - 3s 131ms/step - loss: 5.1320e-04 - root_mean_squared_error: 0.0227
Epoch 15/25
26/26 [==============================] - 4s 136ms/step - loss: 5.5994e-04 - root_mean_squared_error: 0.0237
Epoch 16/25
26/26 [==============================] - 4s 135ms/step - loss: 5.6430e-04 - root_mean_squared_error: 0.0238
Epoch 17/25
26/26 [==============================] - 4s 143ms/step - loss: 5.2251e-04 - root_mean_squared_error: 0.0229
Epoch 18/25
26/26 [==============================] - 4s 136ms/step - loss: 5.7024e-04 - root_mean_squared_error: 0.0239
Epoch 19/25
26/26 [==============================] - 4s 142ms/step - loss: 5.3728e-04 - root_mean_squared_error: 0.0232
Epoch 20/25
26/26 [==============================] - 4s 140ms/step - loss: 5.7911e-04 - root_mean_squared_error: 0.0241
Epoch 21/25
26/26 [==============================] - 4s 143ms/step - loss: 5.8130e-04 - root_mean_squared_error: 0.0241
Epoch 22/25
26/26 [==============================] - 3s 132ms/step - loss: 6.5378e-04 - root_mean_squared_error: 0.0256
Epoch 23/25
26/26 [==============================] - 3s 130ms/step - loss: 5.5375e-04 - root_mean_squared_error: 0.0235
Epoch 24/25
26/26 [==============================] - 3s 132ms/step - loss: 5.3872e-04 - root_mean_squared_error: 0.0232
Epoch 25/25
26/26 [==============================] - 4s 150ms/step - loss: 5.6192e-04 - root_mean_squared_error: 0.0237
```

Out[16]:
```
<keras.src.callbacks.History at 0x1d87f77a340>
```

In [17]:
```
model.evaluate(xtest,ytest)
```

```
4/4 [==============================] - 1s 38ms/step - loss: 0.0026 - root_mean_squared_error: 0.0506
```

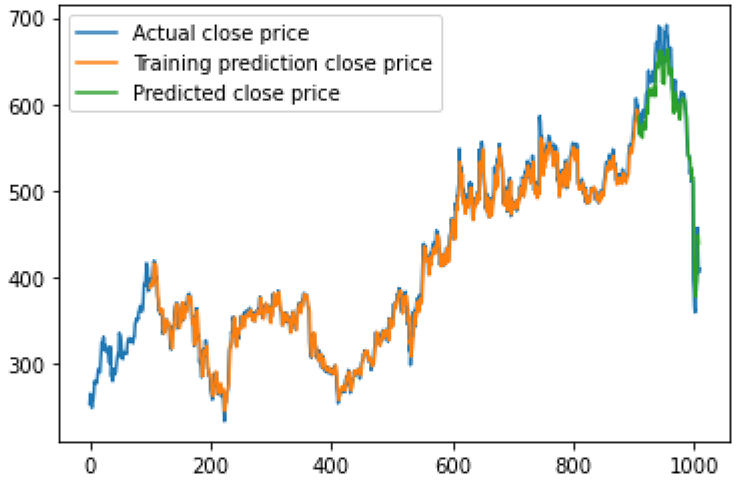
Out[17]:
```
[0.002562620909884572, 0.05062233656644821]
```

In [18]:
```
trainPred=scaler.inverse_transform(model.predict(xtrain)).squeeze()
testPred=scaler.inverse_transform(model.predict(xtest)).squeeze()
```

```
26/26 [==============================] - 2s 55ms/step
4/4 [==============================] - 0s 37ms/step
```

In [19]:
```
look_back=time_step
trainPredPlot=np.empty_like(df)
trainPredPlot[:]=np.nan
trainPredPlot[look_back:len(trainPred)+look_back]=trainPred
testPredPlot=np.empty_like(df)
testPredPlot[:]=np.nan
testPredPlot[len(trainPred)+look_back:len(trainPred)+look_back+len(testPred)]=testPred

plt.plot(df,label="Actual close price")
plt.plot(trainPredPlot,label="Training prediction close price")
plt.plot(testPredPlot,label="Predicted close price")
plt.legend()
plt.show()
```

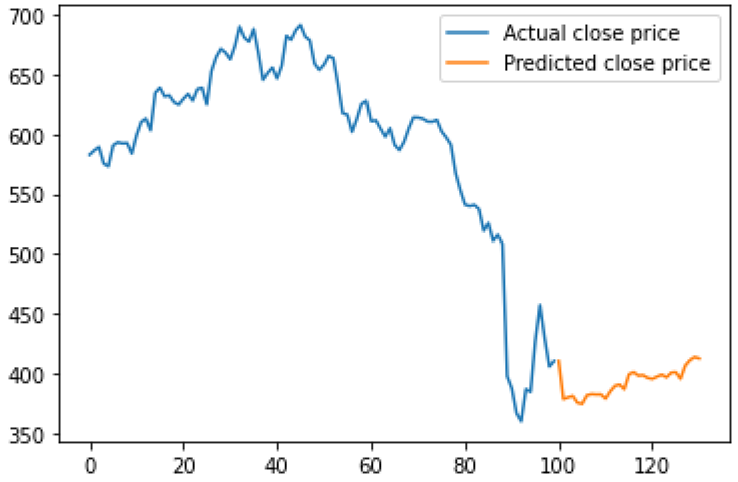


In [20]:
```
input_data=np.array(df[-time_step:])
input_data=input_data.reshape([input_data.shape[0],1])
```

In [21]:
```
def predict(data,days=30):
    data=scaler.transform(data)
    predictions=[]
    i=1
    while(i<=days):
        nxtday=model.predict([data],verbose=0)
        predictions.append(scaler.inverse_transform(nxtday)[0])
        data[:-1]=data[1:]
        data[-1]=nxtday[0]
        i+=1
    return np.array(predictions).squeeze()
```

In [22]:
```
days=30
predictions=predict(input_data,days)
```

In [23]:
```
trainPredPlot=np.zeros(shape=[len(input_data)+1+days])
trainPredPlot[:]=np.nan
trainPredPlot[len(input_data)]=input_data[-1]
trainPredPlot[len(input_data)+1:]=predictions
df_=input_data
plt.plot(df_,label="Actual close price")
plt.plot(trainPredPlot,label="Predicted close price")
plt.legend()
plt.show()
```



In [ ]: