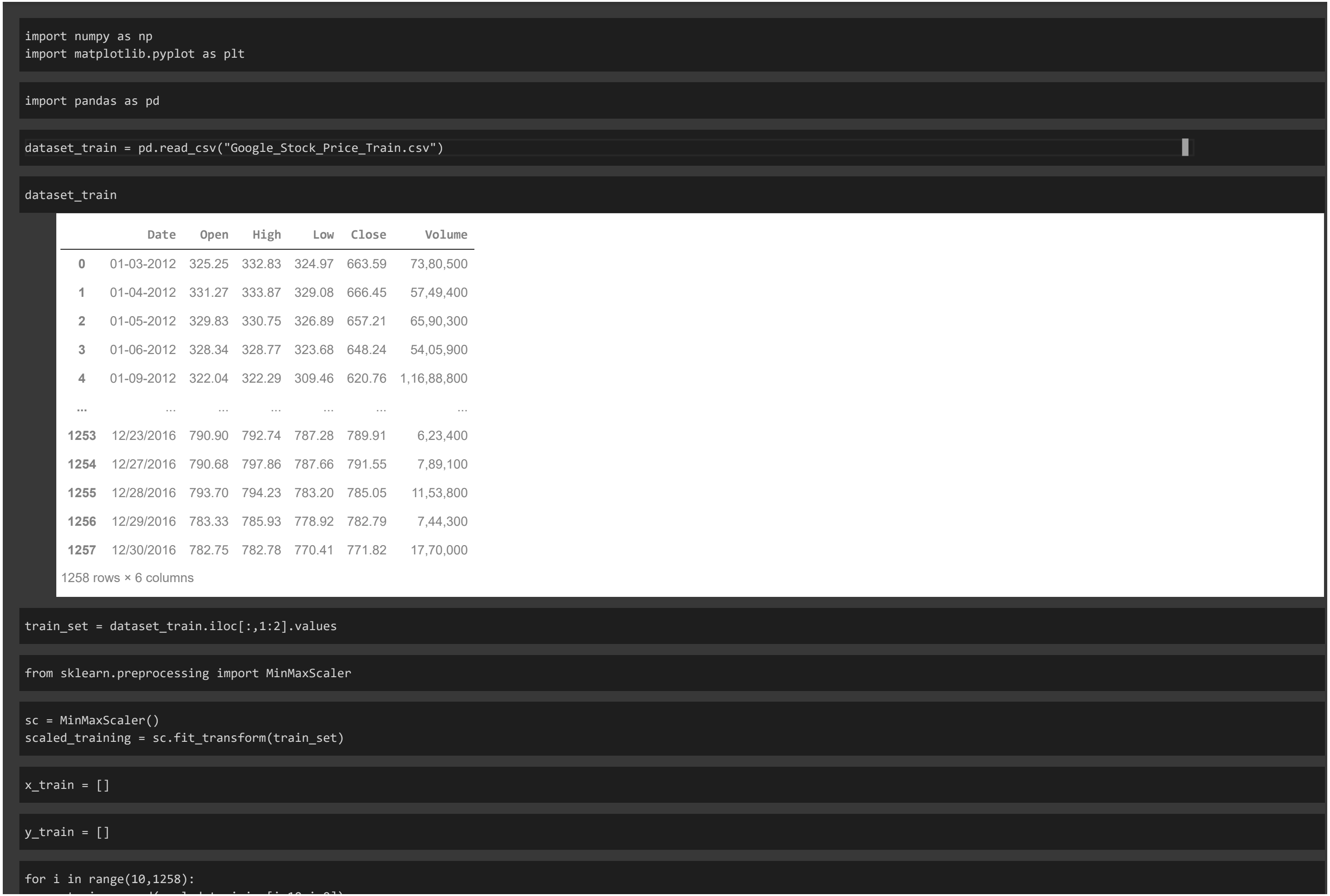


```
import numpy as np
import matplotlib.pyplot as plt
```

```
import pandas as pd
```

```
dataset_train = pd.read_csv("Google_Stock_Price_Train.csv")
```

```
dataset_train
```

	Date	Open	High	Low	Close	Volume
0	01-03-2012	325.25	332.83	324.97	663.59	73,80,500
1	01-04-2012	331.27	333.87	329.08	666.45	57,49,400
2	01-05-2012	329.83	330.75	326.89	657.21	65,90,300
3	01-06-2012	328.34	328.77	323.68	648.24	54,05,900
4	01-09-2012	322.04	322.29	309.46	620.76	1,16,88,800
...	...	...	...	...	...	...
1253	12/23/2016	790.90	792.74	787.28	789.91	6,23,400
1254	12/27/2016	790.68	797.86	787.66	791.55	7,89,100
1255	12/28/2016	793.70	794.23	783.20	785.05	11,53,800
1256	12/29/2016	783.33	785.93	778.92	782.79	7,44,300
1257	12/30/2016	782.75	782.78	770.41	771.82	17,70,000

1258 rows × 6 columns

```
train_set = dataset_train.iloc[:,1:2].values
```

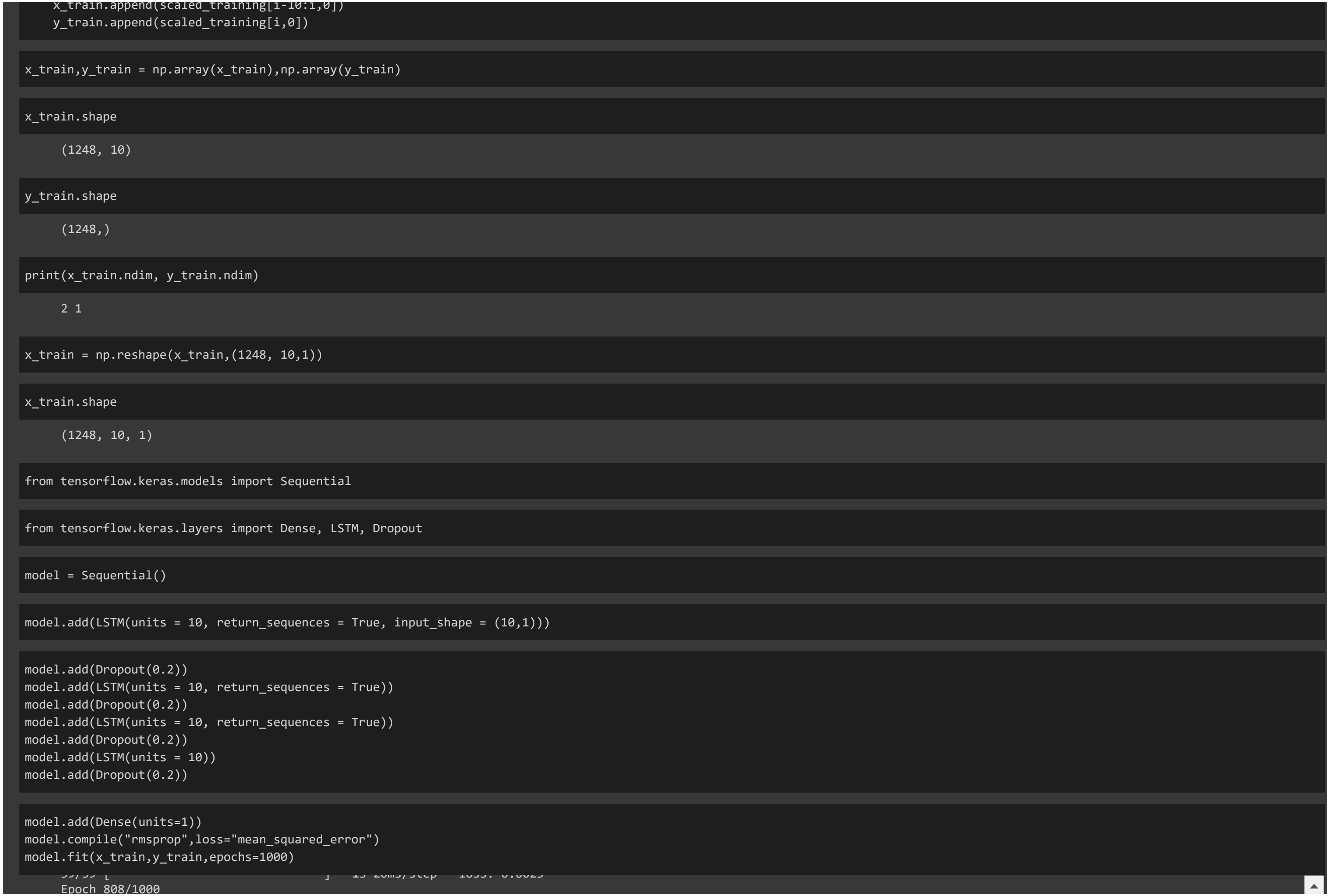
```
from sklearn.preprocessing import MinMaxScaler
```

```
sc = MinMaxScaler()
scaled_training = sc.fit_transform(train_set)
```

```
x_train = []
```

```
y_train = []
```

```
for i in range(10,1258):
```



```
    x_train.append(scaled_training[i-10:i,0])
    y_train.append(scaled_training[i,0])
```

```
x_train,y_train = np.array(x_train),np.array(y_train)
```

```
x_train.shape
```

```
    (1248, 10)
```

```
y_train.shape
```

```
    (1248,)
```

```
print(x_train.ndim, y_train.ndim)
```

```
    2 1
```

```
x_train = np.reshape(x_train,(1248, 10,1))
```

```
x_train.shape
```

```
    (1248, 10, 1)
```

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense, LSTM, Dropout
```

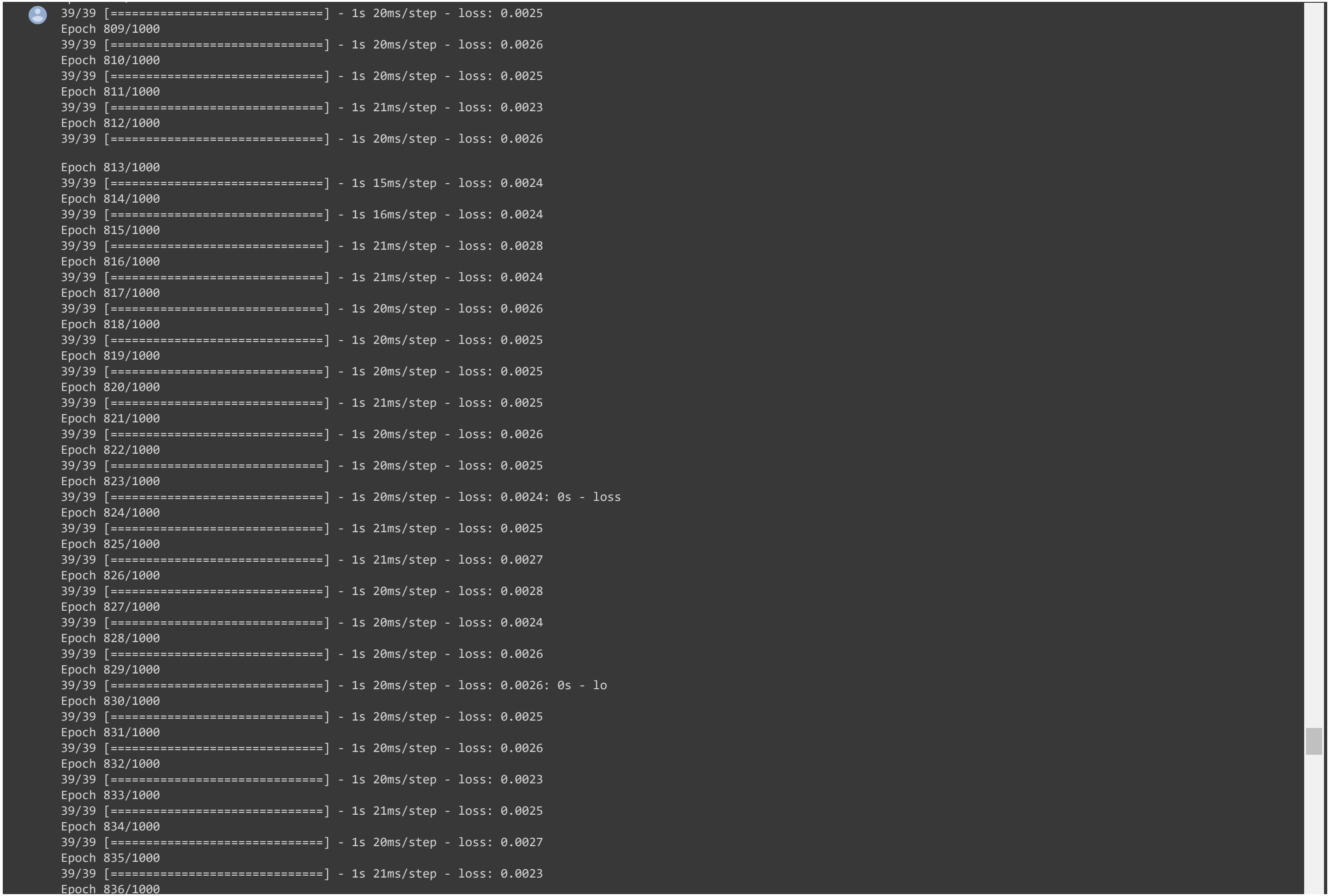
```
model = Sequential()
```

```
model.add(LSTM(units = 10, return_sequences = True, input_shape = (10,1)))
```

```
model.add(Dropout(0.2))
model.add(LSTM(units = 10, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(units = 10, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(units = 10))
model.add(Dropout(0.2))
```

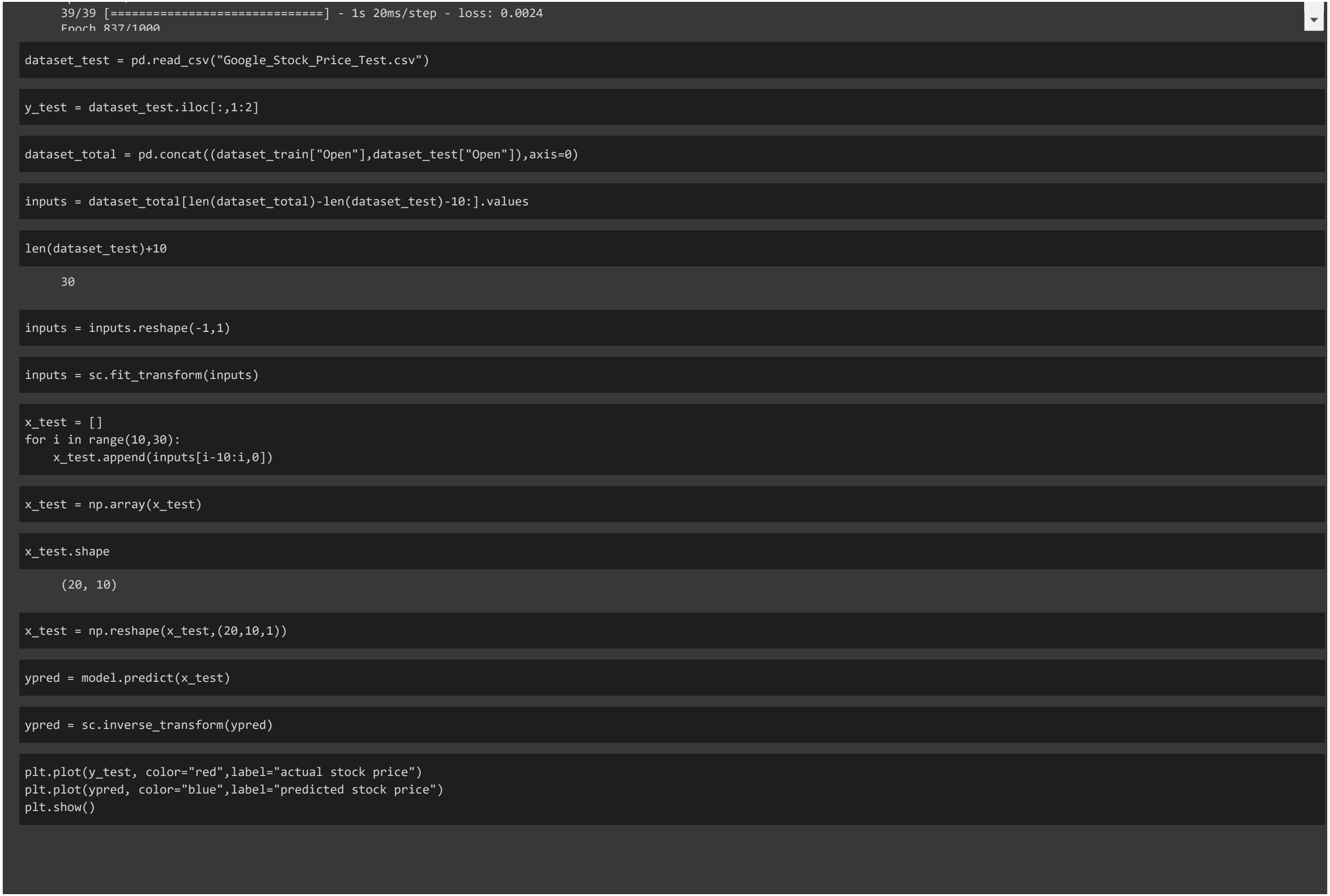
```
model.add(Dense(units=1))
model.compile("rmsprop",loss="mean_squared_error")
model.fit(x_train,y_train,epochs=1000)
```

```
    39/39 [                                ] - 1s 20ms/step - loss: 0.0025
    Epoch 808/1000
```



```
39/39 [==============================] - 1s 20ms/step - loss: 0.0025
Epoch 809/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0026
Epoch 810/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0025
Epoch 811/1000
39/39 [==============================] - 1s 21ms/step - loss: 0.0023
Epoch 812/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0026

Epoch 813/1000
39/39 [==============================] - 1s 15ms/step - loss: 0.0024
Epoch 814/1000
39/39 [==============================] - 1s 16ms/step - loss: 0.0024
Epoch 815/1000
39/39 [==============================] - 1s 21ms/step - loss: 0.0028
Epoch 816/1000
39/39 [==============================] - 1s 21ms/step - loss: 0.0024
Epoch 817/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0026
Epoch 818/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0025
Epoch 819/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0025
Epoch 820/1000
39/39 [==============================] - 1s 21ms/step - loss: 0.0025
Epoch 821/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0026
Epoch 822/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0025
Epoch 823/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0024: 0s - loss
Epoch 824/1000
39/39 [==============================] - 1s 21ms/step - loss: 0.0025
Epoch 825/1000
39/39 [==============================] - 1s 21ms/step - loss: 0.0027
Epoch 826/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0028
Epoch 827/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0024
Epoch 828/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0026
Epoch 829/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0026: 0s - lo
Epoch 830/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0025
Epoch 831/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0026
Epoch 832/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0023
Epoch 833/1000
39/39 [==============================] - 1s 21ms/step - loss: 0.0025
Epoch 834/1000
39/39 [==============================] - 1s 20ms/step - loss: 0.0027
Epoch 835/1000
39/39 [==============================] - 1s 21ms/step - loss: 0.0023
Epoch 836/1000
```



```
39/39 [==============================] - 1s 20ms/step - loss: 0.0024
Epoch 837/1000
```

```
dataset_test = pd.read_csv("Google_Stock_Price_Test.csv")
```

```
y_test = dataset_test.iloc[:,1:2]
```

```
dataset_total = pd.concat((dataset_train["Open"],dataset_test["Open"]),axis=0)
```

```
inputs = dataset_total[len(dataset_total)-len(dataset_test)-10:].values
```

```
len(dataset_test)+10
```

```
30
```

```
inputs = inputs.reshape(-1,1)
```

```
inputs = sc.fit_transform(inputs)
```

```
x_test = []
for i in range(10,30):
    x_test.append(inputs[i-10:i,0])
```

```
x_test = np.array(x_test)
```

```
x_test.shape
```

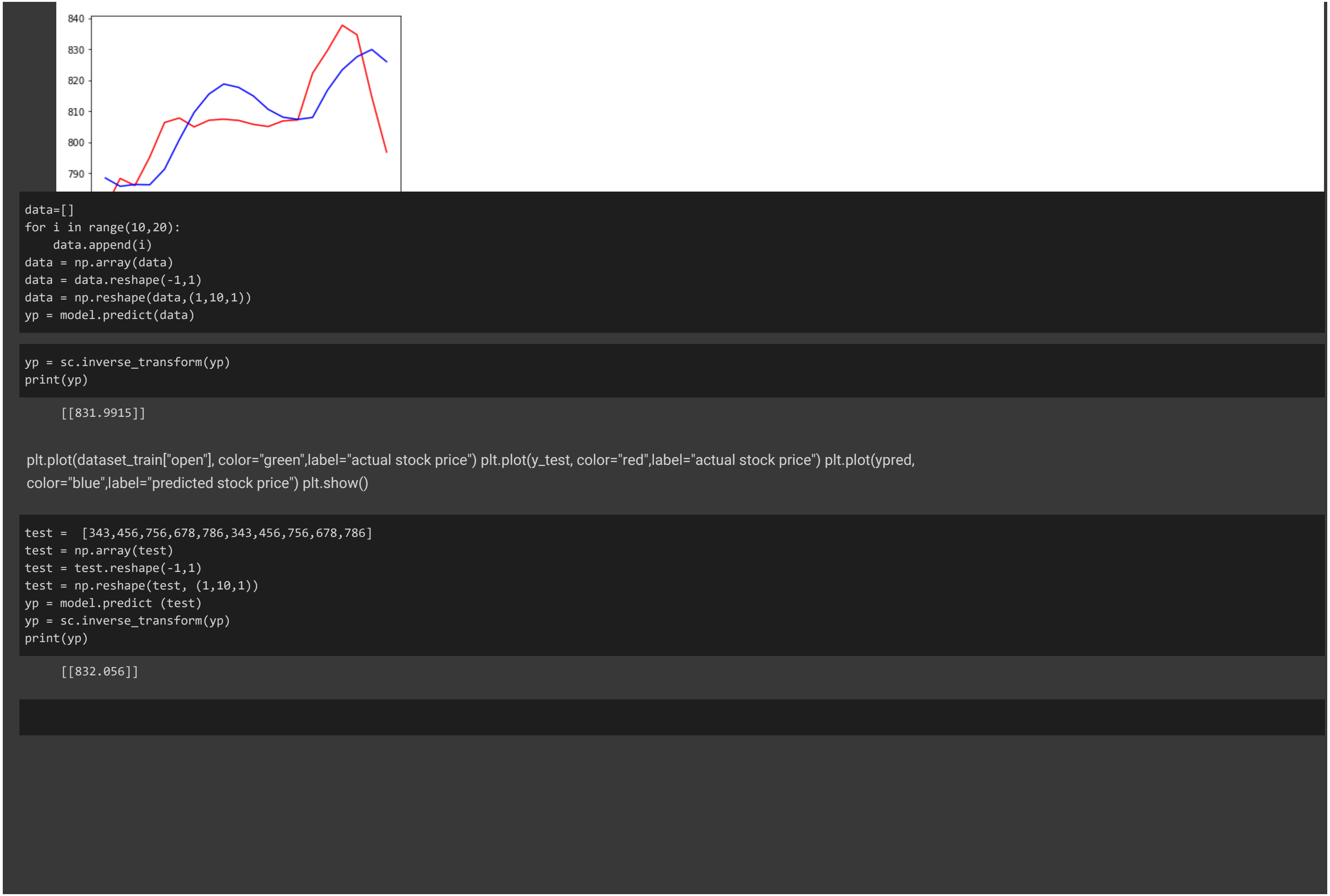
```
(20, 10)
```

```
x_test = np.reshape(x_test,(20,10,1))
```

```
ypred = model.predict(x_test)
```

```
ypred = sc.inverse_transform(ypred)
```

```
plt.plot(y_test, color="red",label="actual stock price")
plt.plot(ypred, color="blue",label="predicted stock price")
plt.show()
```



```
data=[]
for i in range(10,20):
    data.append(i)
data = np.array(data)
data = data.reshape(-1,1)
data = np.reshape(data,(1,10,1))
yp = model.predict(data)
```

```
yp = sc.inverse_transform(yp)
print(yp)
```

```
[[831.9915]]
```

plt.plot(dataset_train["open"], color="green",label="actual stock price") plt.plot(y_test, color="red",label="actual stock price") plt.plot(ypred, color="blue",label="predicted stock price") plt.show()

```
test =  [343,456,756,678,786,343,456,756,678,786]
test = np.array(test)
test = test.reshape(-1,1)
test = np.reshape(test, (1,10,1))
yp = model.predict (test)
yp = sc.inverse_transform(yp)
print(yp)
```

```
[[832.056]]
```