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

In [5]:
```
import torch
import torch.nn as nn
```

In [4]:
```
pip install torch
```

```
Requirement already satisfied: torch in c:\users\dell\anaconda3\lib\site-
packages (2.1.0)
Requirement already satisfied: networkx in c:\users\dell\anaconda3\lib\si
te-packages (from torch) (2.8.4)
Requirement already satisfied: jinja2 in c:\users\dell\anaconda3\lib\site
-packages (from torch) (2.11.3)
Requirement already satisfied: fsspec in c:\users\dell\anaconda3\lib\site
-packages (from torch) (2022.7.1)
Requirement already satisfied: sympy in c:\users\dell\anaconda3\lib\site-
packages (from torch) (1.10.1)
Requirement already satisfied: typing-extensions in c:\users\dell\anacond
a3\lib\site-packages (from torch) (4.3.0)
Requirement already satisfied: filelock in c:\users\dell\anaconda3\lib\si
te-packages (from torch) (3.6.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\dell\anaconda
3\lib\site-packages (from jinja2->torch) (2.0.1)
Requirement already satisfied: mpmath>=0.19 in c:\users\dell\anaconda3\li
b\site-packages (from sympy->torch) (1.2.1)
Note: you may need to restart the kernel to use updated packages.

WARNING: Ignoring invalid distribution -cipy (c:\users\dell\anaconda3\lib
\site-packages)
WARNING: Ignoring invalid distribution -cipy (c:\users\dell\anaconda3\lib
\site-packages)
WARNING: Ignoring invalid distribution -cipy (c:\users\dell\anaconda3\lib
\site-packages)
WARNING: Ignoring invalid distribution -cipy (c:\users\dell\anaconda3\lib
\site-packages)
WARNING: Ignoring invalid distribution -cipy (c:\users\dell\anaconda3\lib
\site-packages)
WARNING: Ignoring invalid distribution -cipy (c:\users\dell\anaconda3\lib
\site-packages)
```

In [6]:
```
df = pd.read_csv("netflix.csv")
closed_prices = df["Close"]
```

In [7]:
```
seq_len = 15
```

In [8]:
```
mm = MinMaxScaler()
scaled_price = mm.fit_transform(np.array(closed_prices)[... , None]).squee
```

In [9]:
```
X = []
```

In [9]:
```
X = []
y = []

for i in range(len(scaled_price) - seq_len):
    X.append(scaled_price[i : i + seq_len])
    y.append(scaled_price[i + seq_len])
```

In [10]:
```
X = np.array(X)[... , None]
y = np.array(y)[... , None]
```

In [11]:
```
train_x = torch.from_numpy(X[:int(0.8 * X.shape[0])]).float()
train_y = torch.from_numpy(y[:int(0.8 * X.shape[0])]).float()
test_x = torch.from_numpy(X[int(0.8 * X.shape[0]):]).float()
test_y = torch.from_numpy(y[int(0.8 * X.shape[0]):]).float()
```

In [12]:
```
class Model(nn.Module):
    def __init__(self , input_size , hidden_size):
        super().__init__()
        self.lstm = nn.LSTM(input_size , hidden_size , batch_first = True)
        self.fc = nn.Linear(hidden_size , 1)
    def forward(self , x):
        output , (hidden , cell) = self.lstm(x)
        return self.fc(hidden[-1 , :])
model = Model(1 , 64)
```

In [13]:
```
optimizer = torch.optim.Adam(model.parameters() , lr = 0.001)
loss_fn = nn.MSELoss()

num_epochs = 100
```

In [14]:
```
for epoch in range(num_epochs):
    output = model(train_x)
    loss = loss_fn(output , train_y)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if epoch % 10 == 0 and epoch != 0:
        print(epoch , "epoch loss" , loss.detach().numpy())
```

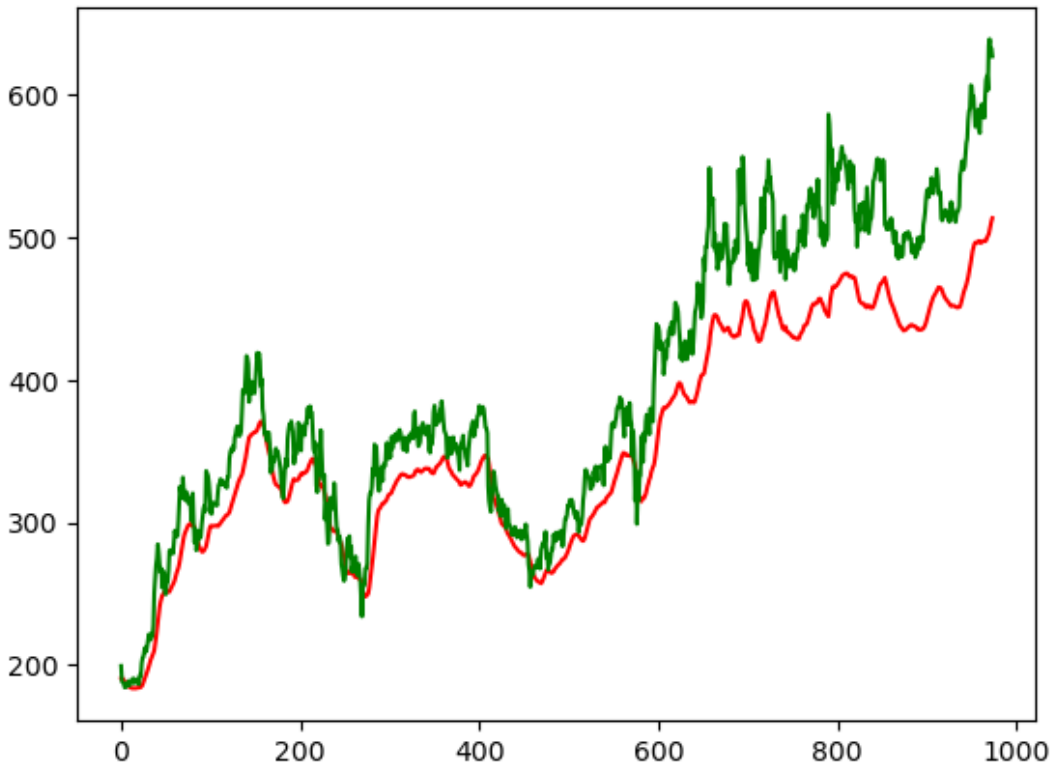
```
10 epoch loss 0.0053171725
20 epoch loss 0.004698529
30 epoch loss 0.0040599126
40 epoch loss 0.0031257484
50 epoch loss 0.001223189
60 epoch loss 0.00044837545
70 epoch loss 0.0001615438
80 epoch loss 2.57842e-05
90 epoch loss 1.9511599e-05
```

In [15]:
```
model.eval()
with torch.no_grad():
    output = model(test_x)
```

In [16]:
```
pred = mm.inverse_transform(output.numpy())
real = mm.inverse_transform(test_y.numpy())
```

In [16]:
```
pred = mm.inverse_transform(output.numpy())
real = mm.inverse_transform(test_y.numpy())
```

In [17]:
```
plt.plot(pred.squeeze() , color = "red" , label = "predicted")
plt.plot(real.squeeze() , color = "green" , label = "real")
plt.show()
```



In [ ]:
```

```