In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
numpy.random.seed(7)
```

In [6]:
```
df = pd.read_csv('airline-passengers.csv', usecols=[1])
```

In [7]:
```
df.head()
```

Out[7]:

	Passengers
0	112
1	118
2	132
3	129
4	121

In [8]:
```
## converting dataframe into numpy array
df = df.values
```

In [9]:
```
## changing the datatype
df = df.astype('float32')
```

In [10]:
```
## Scaling
scaler = MinMaxScaler()
df = scaler.fit_transform(df)
```

In [11]:
```
df
```

Out[11]:
```
array([[0.01544401],
       [0.02702703],
       [0.05405405],
       [0.04826255],
       [0.03281853],
       [0.05984557],
       [0.08494207],
       [0.08494207],
       [0.06177607],
       [0.02895753],
       [0.        ],
       [0.02702703],
       [0.02123553],
       [0.04247104],
       [0.07142857],
       [0.05984557],
       [0.04054055],
       [0.08687258],
       [0.12741312],
```

In [13]:
```
train_size = int(len(df) * 0.67)
test_size = len(df)-train_size
train,test = df[0:train_size,:], df[train_size:len(df)]
print(len(train), len(test))
```

```
96 48
```

In [14]:
```
# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return numpy.array(dataX), numpy.array(dataY)
```

In [15]:
```
look_back = 1
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
# reshape input to be [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

In [16]:
```
## model
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=100, batch_size=1, verbose=2)
```

```
94/94 - 0s - 2ms/step - loss: 0.0028
Epoch 16/100
94/94 - 0s - 1ms/step - loss: 0.0024
Epoch 17/100
94/94 - 0s - 1ms/step - loss: 0.0024
Epoch 18/100
94/94 - 0s - 1ms/step - loss: 0.0021
Epoch 19/100
94/94 - 0s - 1ms/step - loss: 0.0021
Epoch 20/100
94/94 - 0s - 2ms/step - loss: 0.0021
Epoch 21/100
94/94 - 0s - 1ms/step - loss: 0.0021
Epoch 22/100
94/94 - 0s - 1ms/step - loss: 0.0021
Epoch 23/100
94/94 - 0s - 1ms/step - loss: 0.0020
Epoch 24/100
94/94 - 0s - 1ms/step - loss: 0.0021
Epoch 25/100
```

In [17]:
```
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

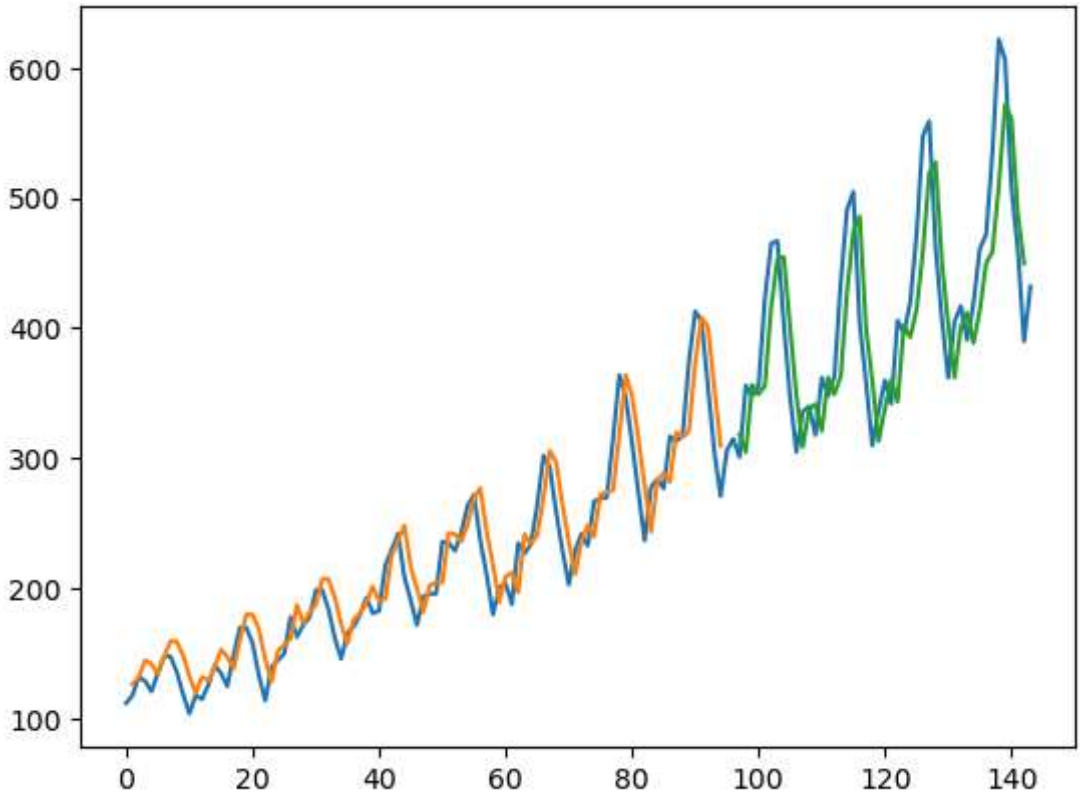
In [18]:
```
# invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
```

In [19]:
```
# calculate root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
```

In [22]:
```
# shift train predictions for plotting
trainPredictPlot = numpy.empty_like(df)
trainPredictPlot[:, :] = numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, :] = trainPredict
```

In [25]:
```
# shift test predictions for plotting
testPredictPlot = numpy.empty_like(df)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(df)-1, :] = testPredict
```

In [26]:
```
# plot baseline and predictions
plt.plot(scaler.inverse_transform(df))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```



In [ ]: