```
# Program to implement Simple RNN Model
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, SimpleRNN

# convert into dataset matrix
def convertToMatrix(data, step):
 X, Y =[], []
 for i in range(len(data)-step):
  d=i+step
  X.append(data[i:d,])
  Y.append(data[d,])
 return np.array(X), np.array(Y)

step = 4
N = 1000
Tp = 800
#Tp = 200

t=np.arange(0,N)
#x=np.sin(0.02*t)+2*np.random.rand(N)
x=np.cos(0.02*t) #+2*np.random.rand(N)
df = pd.DataFrame(x)
df.head()

plt.plot(df)
plt.show()

values=df.values
train,test = values[0:Tp,:], values[Tp:N,:]

# add step elements into train and test
test = np.append(test,np.repeat(test[-1,],step))
train = np.append(train,np.repeat(train[-1,],step))

trainX,trainY =convertToMatrix(train,step)
testX,testY =convertToMatrix(test,step)
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

model = Sequential()
#model.add(SimpleRNN(units=32, input_shape=(1,step),
activation="relu"))
model.add(SimpleRNN(units=16, input_shape=(1,step), activation="tanh"))
model.add(Dense(8, activation="relu"))
model.add(Dense(1))
```

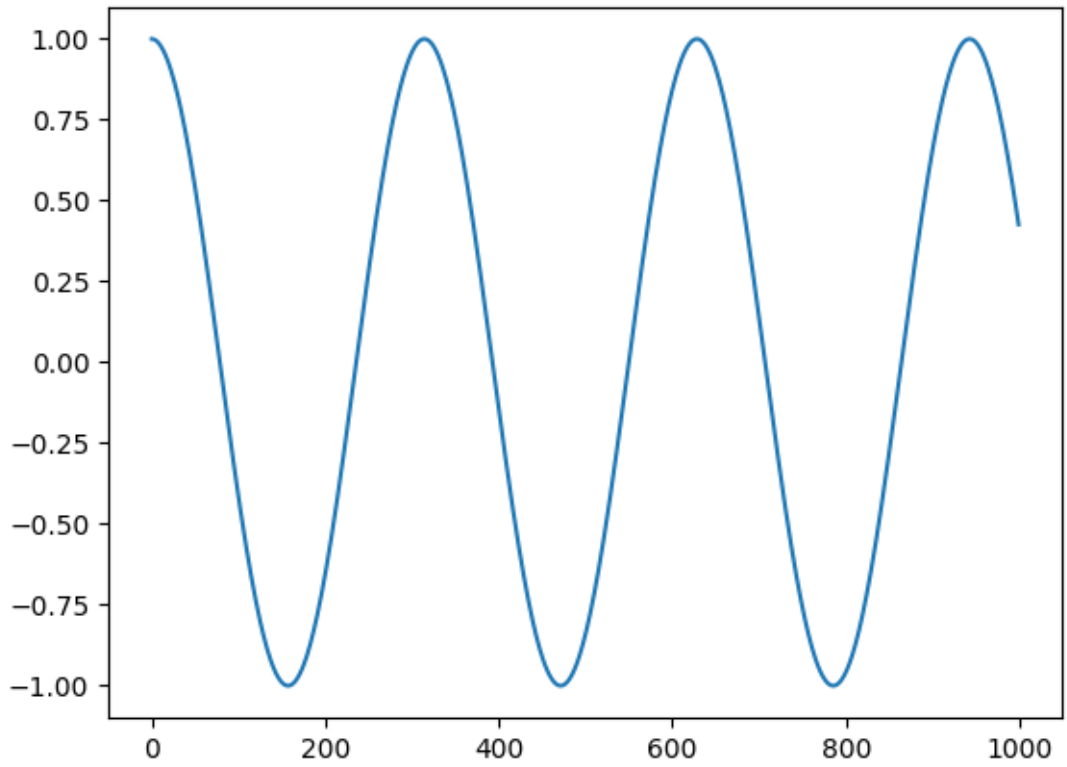
```
#model.compile(loss='mean_squared_error', optimizer='rmsprop')
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()

#model.fit(trainX,trainY, epochs=100, batch_size=16, verbose=2)
#model.fit(trainX,trainY, epochs=5, batch_size=16)
model.fit(trainX,trainY, epochs=5, batch_size=16)
trainPredict = model.predict(trainX)
testPredict= model.predict(testX)
predicted=np.concatenate((trainPredict,testPredict),axis=0)

#trainScore = model.evaluate(trainX, trainY, verbose=0)
#print(trainScore)

index = df.index.values
plt.plot(index,df)
plt.plot(index,predicted)
plt.axvline(df.index[Tp], c="g")
plt.show(
```

OUTPUT




```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 simple_rnn_4 (SimpleRNN)     (None, 16)                336
```

```
 dense_8 (Dense)             (None, 8)                 136

 dense_9 (Dense)             (None, 1)                 9

=================================================================
Total params: 481 (1.88 KB)
Trainable params: 481 (1.88 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/5
50/50 [==============================] - 1s 2ms/step - loss: 0.1480
Epoch 2/5
50/50 [==============================] - 0s 3ms/step - loss: 0.0235
Epoch 3/5
50/50 [==============================] - 0s 3ms/step - loss: 0.0028
Epoch 4/5
50/50 [==============================] - 0s 3ms/step - loss: 0.0012
Epoch 5/5
50/50 [==============================] - 0s 3ms/step - loss: 7.5570e-04
25/25 [==============================] - 0s 2ms/step
7/7 [==============================] - 0s 2ms/step
```