In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense,SimpleRNN
```

In [23]:
```
# Generate sample data

N=1000
Tp=800
t=np.random.rand(N)
df=pd.DataFrame(t)
df
```

Out[23]:

	0
0	0.125009
1	0.816485
2	0.395263
3	0.486627
4	0.762993
...	...
995	0.673239
996	0.145410
997	0.373292
998	0.293297
999	0.894439

1000 rows × 1 columns

In [16]:
```
# Splitting data
values=df.values
train,test=values[0:Tp,:],values[Tp:N,:]
```

In [22]:
```
step=4
train=np.append(train,np.repeat(train[-1,],step))
test=np.append(test,np.repeat(test[-1,],step))
```

In [32]:
```
# converting 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)

train_x,train_y=convertToMatrix(train,step)
test_x,test_y=convertToMatrix(test,step)


train_x.shape
```

Out[32]:
```
(804, 4)
```

In [33]:
```
# Reshape train_x,test_x to fit keras model.Since RNN model requires 3D input data
train_x=np.reshape(train_x,(train_x.shape[0],1,train_x.shape[1]))
test_x=np.reshape(test_x,(test_x.shape[0],1,test_x.shape[1]))
train_x.shape
```

Out[33]:
```
(804, 1, 4)
```

In [34]:
```
# Building a model using simple RNN model

model=Sequential()
model.add(SimpleRNN(units=32,input_shape=(1,step),activation='relu'))
model.add(Dense(8,activation='relu'))
model.add(Dense(1))
```

In [35]:
```
# Minimizing loss using optimizer
model.compile(loss='mean_squared_error',optimizer='rmsprop')
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_1 (SimpleRNN)     (None, 32)                1184
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 264
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 9
=================================================================
Total params: 1,457
Trainable params: 1,457
Non-trainable params: 0
_________________________________________________________________
```

In [44]:
```
model.fit(train_x,train_y,epochs=100,batch_size=16,verbose=1)
```

```
Epoch 90/100
51/51 [==============================] - 0s 2ms/step - loss: 0.0806
Epoch 91/100
51/51 [==============================] - 0s 2ms/step - loss: 0.0805
Epoch 92/100
51/51 [==============================] - 0s 2ms/step - loss: 0.0803
Epoch 93/100
51/51 [==============================] - 0s 2ms/step - loss: 0.0805
Epoch 94/100
51/51 [==============================] - 0s 2ms/step - loss: 0.0806
Epoch 95/100
51/51 [==============================] - 0s 2ms/step - loss: 0.0807
Epoch 96/100
51/51 [==============================] - 0s 2ms/step - loss: 0.0803
Epoch 97/100
51/51 [==============================] - 0s 2ms/step - loss: 0.0800
Epoch 98/100
51/51 [==============================] - 0s 2ms/step - loss: 0.0807
Epoch 99/100
```

In [45]:

In [46]:
```
# check the loss
train_score=model.evaluate(test_x,test_y,verbose=0)
print(train_score)
```

```
0.09313173592090607
```

In [ ]: