In [1]:

```
# import libraries

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN
from tensorflow.keras.layers import Dense

import numpy as np
from numpy import array
import os
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
# define input sequence

raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]

# choose a number of time steps
seq_len = 3

# choose number of future time steps to predict
n_steps = 2
```

In [3]:

```
def split_sequence(sequence, seq_len, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + seq_len
        # check if we are beyond the sequence
        if end_ix + n_steps > len(sequence):
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix:end_ix+n_steps]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)
```

In [4]:

```
# split into sequences

seq_X, seq_Y = split_sequence(raw_seq, seq_len, n_steps)

# summarize the data
for i in range(len(seq_X)):
    print(seq_X[i], seq_Y[i])
```

```
[10 20 30] [40 50]
[20 30 40] [50 60]
[30 40 50] [60 70]
[40 50 60] [70 80]
[50 60 70] [80 90]
```

In [5]:

```
# split the sequence generated into final training X and Y sequence to be fed to RNN model

X = seq_X.reshape((seq_X.shape[0], seq_X.shape[1], 1))
Y = seq_Y


print(X.shape, Y.shape)
```

```
(5, 3, 1) (5, 2)
```

In [8]:

```
#RNN
model = Sequential()
model.add(SimpleRNN(50, activation='relu', input_shape=(seq_len, 1)))
model.add(Dense(n_steps))
model.compile(optimizer='adam', loss='mse')

print(model.summary())
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
simple_rnn_1 (SimpleRNN)     (None, 50)                2600      
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 102       
=================================================================
Total params: 2,702
Trainable params: 2,702
Non-trainable params: 0
_________________________________________________________________
None
```

For Simple RNN layer:

Number of parameters is given by the formula:

(num_features + num_units)* num_units + num_units

num_units = equals the number of units in the RNN

num_features = equals the number features of your input

Now you have two things happening in your RNN.

First you have the recurrent loop, where the state is fed recurrently into the model to generate the next step. Weights for the recurrent step are:

recurrent_weights = num_units*num_units

The secondly you have new input of your sequence at each step.

input_weights = num_features*num_units

So now we have the weights, whats missing are the biases - for every unit one bias:

biases = num_units*1

So finally we have the formula:

(num_features + num_units)* num_units + num_units = (50 + 1) * 50 + 50 = 2600

For Output layer:

**output_channel_number * (input_channel_number + 1) = 2 * (50 + 1) =102

In [9]:

```
# model training

history = model.fit(X, Y, epochs= 200, verbose=0)
```

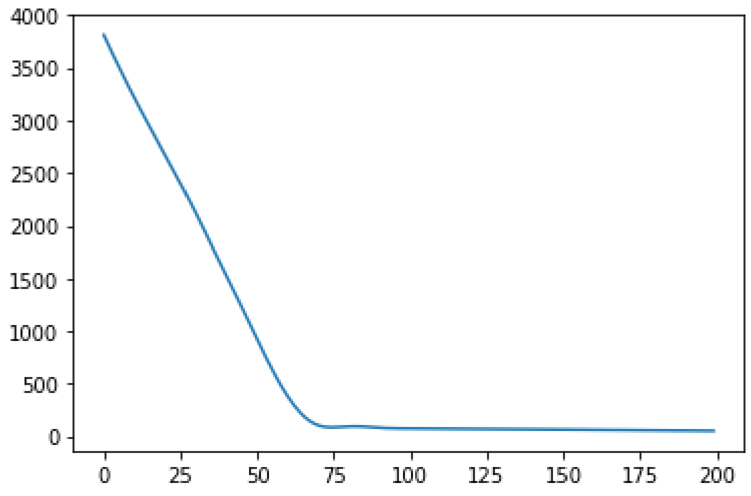
In [12]:

```
# plot training loss

plt.plot(history.history["loss"])
```

Out[12]:

```
[<matplotlib.lines.Line2D at 0x7f57b9dcf2e8>]
```



In [11]:

```
# Test Prediction

X_test = array([70, 80, 90])
X_test = X_test.reshape((1, seq_len, 1))
Y_test = model.predict(X_test)
print(Y_test)
```

```
[[114.02077 127.01842]]
```

In [10]:

```
keras.backend.clear_session()
```