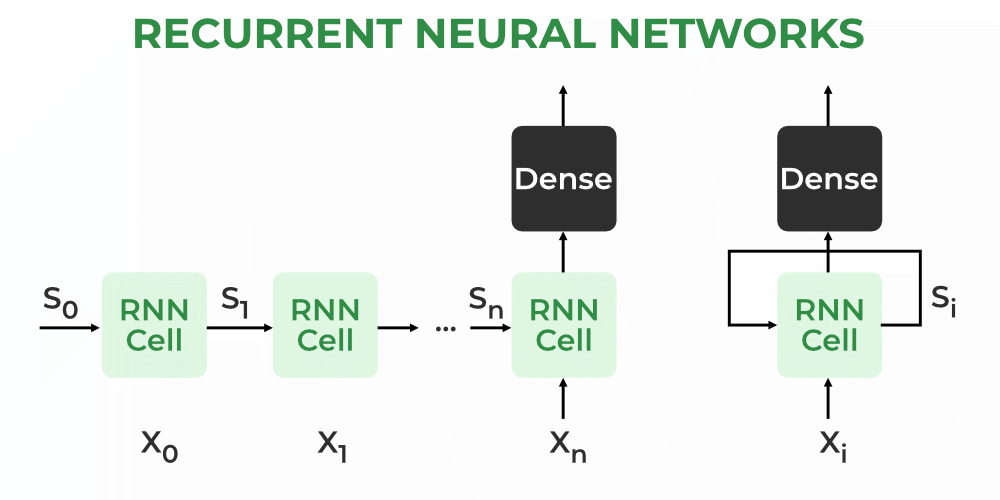
Experiment-4

Aim:- Implementation of RNN(Recurrent Neural Network).

RNN:- It is a type of artificial neural network designed for processing sequences of data. It is use for task that involves time series data. RNN has recurrent connection, which allow them to maintain a hidden state or memory of previous time steps and use it in current input.



Source:- Geeks For Geeks

RNN consist of multiple fixed activation function unit, one is for each time step. In this hidden layer signifies the past knowledge that network currently hold at given time step. We mostly use tanh function here.



**Code:-**

|  |
| --- |
| import numpy as np  input\_size=3  hidden\_size=4  sequence\_length=5 #tell how many time we have to set back  #random select weight for input layer to hidden layer  wxh=np.random.randn(hidden\_size, input\_size)  #random select weight for hiden layer to hidden  whh = np.random.randn(hidden\_size, hidden\_size)  #bias for hidden layer  bh=np.zeros((hidden\_size,1))  #next hidden state or we can say previous hidden state because all previous data is going to be  h\_prev=np.zeros((hidden\_size,1))  #forward pass  def rnn\_forward(x, h\_prev, Wxh, Whh, bh):  # Calculate hidden state  h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h\_prev) + bh)    return h  #Now generate the input  x = np.random.randn(input\_size, sequence\_length)  #forward pass for each time stamp  hidden\_states = []  for t in range(sequence\_length):  h\_prev = rnn\_forward(x[:, t:t+1], h\_prev, wxh, whh, bh)  hidden\_states.append(h\_prev)  #print hidden states  for t, h in enumerate(hidden\_states):  print(f"Time Step {t+1}:")  print(h) |

**Output:-**

