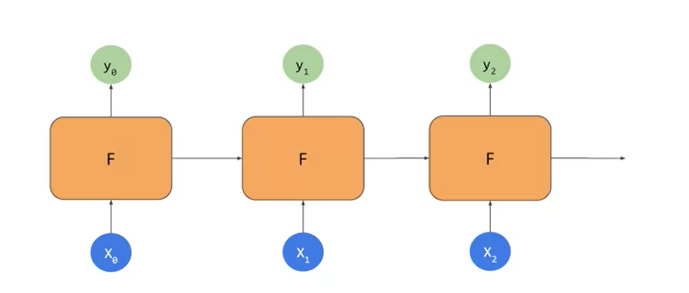
The default activation function in the RNN layers is tan H which is the hyperbolic tangent activation.

an RNN expects three-dimensions; batch size, the number of timestamps,

and the series dimensionality. With the Lambda layer, we can fix this without rewriting our Window dataset helper function

***RNN***



but perhaps a better approach would be to use LSTMs instead of RNNs to see the impact. We'll explore that in this video. If you remember when you looked at RNNs, they looked a little bit like this. They had cells that took patches as inputs or X, and they calculated a Y output as well as the state vector, that fed into the cell along with the next X which then resulted in the Y, and the state vector and so on. The impact of this is that while state is a factor in subsequent calculations, its impacts can diminish greatly over timestamps. LSTMs are the cell state to this that keep a state throughout the life of the training so that the state is passed from cell to cell, timestamp to timestamp, and it can be better maintained. This means that the data from earlier in the window can have

a greater impact on the overall projection than in the case of RNNs. The state can also be bidirectional so that state moves forwards and backwards. In the case of texts, this was really powerful. Within the prediction of numeric sequences, it may or may not be, and it'll be interesting to experiment with. I'm not going to go into a lot of detail here but hundreds videos around LSTM are terrific. From there, you can really understand how they work under the hood.

