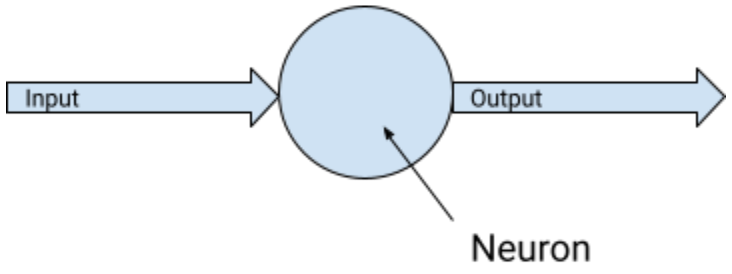
More on Neural Networks

Up to now you've been looking at matching X values to Y values when there's a linear relationship between them. So, for example, you matched the X values in [-1, 0 , 1, 2, 3, 4] to the Y values in [-3, -1, 1, 3, 5, 7] by figuring out the equation Y=2X-1.

You then saw how a very simple neural network with a single neuron could be used for this.



This worked very well, because, in reality, what is referred to as a 'neuron' here is simply a function that has two learnable parameters, called a 'weight' and a 'bias', where, the output of the neuron will be:

Output = (Weight * Input) + Bias

So, for learning the linear relationship between our Xs and Ys, this maps perfectly, where we want the weight to be learned as '2', and the bias as '-1'. In the code you saw this happening.

When multiple neurons work together in layers, the learned weights and biases across these layers can then have the effect of letting the neural network learn more complex patterns. You'll learn more about how this works later in the course.

In your first Neural Network you saw neurons that were densely connected to each other, so you saw the **Dense** layer type. As well as neurons like this, there are also additional layer types in TensorFlow that you'll encounter. Here's just a few of them:

- **Convolutional** layers contain filters that can be used to transform data. The values of these filters will be learned in the same way as the parameters in the Dense neuron you saw here. Thus, a network containing them can learn how to transform data effectively. This is especially useful in Computer Vision, which you'll see later in this course. We'll even use these convolutional layers that are typically used for vision models to do speech detection! Are you wondering how or why? Stay tuned!
- **Recurrent** layers learn about the relationships between pieces of data in a sequence. There are many types of recurrent layer, with a popular one called LSTM (Long, Short Term Memory), being particularly effective. Recurrent layers are useful for predicting sequence data (like the weather), or understanding text.

You'll also encounter layer types that *don't* learn parameters themselves, but which can affect the other layers. These include layers like **dropouts**, which are used to reduce the density of connection between dense layers to make them more efficient, **pooling** which can be used to reduce the amount of data flowing through the network to remove unnecessary information, and **lambda** layers that allow you to execute arbitrary code.

Your journey over the next few videos will primarily deal with the **Dense** network type, and you'll start to explore how multiple layers can work together to infer the rules that match your data to your labels.