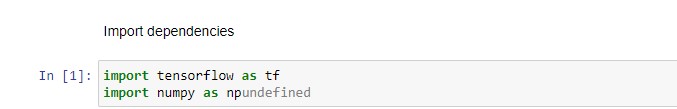
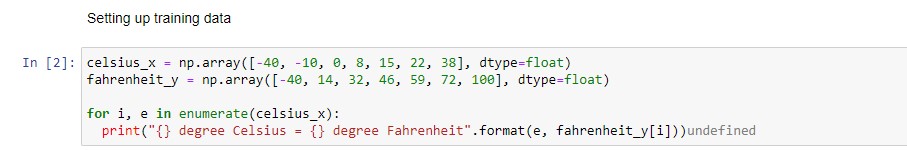
**Import dependencies**

First import TensorFlow. Here we are calling tf for ease of use. We also tell it to only display errors. Next, import NumPy as np that helps us to represent our data as highly performant lists.



**Set up training data**:

Supervised Machine Learning is all about figuring out an algorithm given a set of inputs and outputs. since the task in this colab is to create a model that can give the temperature in Fahrenheit when given the degree in Celsius, we create two lists celsius\_x and fahrenheit\_y that we can use to train our model.



**Some Machine Learning Terminology**

Feature: The input(s) to our model. In this case, a single value - the degree in Celsius. Labels: The output our model predicts. In this case, a single value - the degree in Fahrenheit.

**Example:** A pair of i/o used during training. In our case a pair of values from celsius\_x and farhenheit\_y at a specific index, such as (22, 72).

**Create the model:**

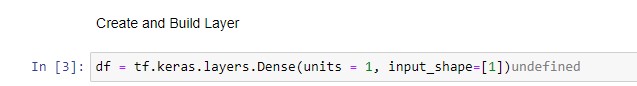
Next create the model. We will use simplest possible model we can, a Dense network. Since the problem is straightforward, this network will require only a single layer, with a single neuron.

Build a layer

We'll call the layer 10 and create it by instantiating tf. keras.layers.Dense with the following configuration:

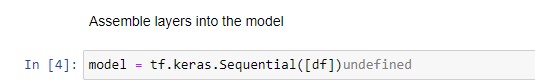
**input\_shape** = [1] - This specifies that the input to this layer is a single value. That is, the shape is one-dimensional array with one member. Since this is the first (and only) layer, that input shape is the input shape of the entire model. The single value is a floating-point number, representing degrees Celsius.

**units** =1 - This specifies the number of neurons in the layer. The number of neurons defines how many internal variables the layer has to try to learn how to solve the problem (more later). Since this is the final layer, it is also the size of the model's output - a single float value representing degrees Fahrenheit. (In a multi-layered network, the size and shape of the later would need to match the input\_shape of the next layer.)



**Assemble layers into the model:**

Once layers are defined, they need to be assembled into a model. The Sequential model definition takes a list of layers as argument, specifying the calculation order from the input to the output. This model has just a single layer.



**Compile the model, with loss and optimizer functions**:

before training, the model has to be compiled. When compiled for training, the model is given:

**Loss function**: A way of measuring how far off predictions are from the desired outcome. (The measure difference is called the "loss")

**Optimizer function**: A way of adjusting internal values in order to reduce the loss.

