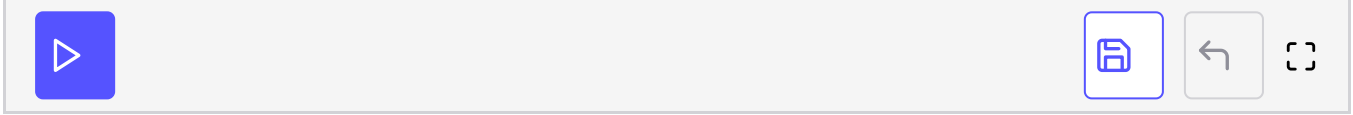
Testing Our Code Thus Far

In this lesson, we will test our code with some random inputs and see if our neural network is built properly.

Before we go on to write the `train()` function for training the network with examples, let's just test the code we have at this point works. Let's create a small network and query it with some random input, just to see it working. Obviously, there won't be any real meaning to this; we're only doing this to use these functions we just created.

The following shows the creation of a small network with three nodes in each of the input, hidden and output layers, and queries it with a randomly chosen input of $(1.0, 0.5, -1.5)$

```
#number of input, hidden and ou
input_nodes= 3
hidden_nodes = 3
output_nodes = 3
# learning rate is 0.3
learning_rate = 0.3
#create instance of nueral netw
n = neuralNetwork(input_nodes,h
#query it with random inputs an
output = n.query([1.0,0.5,-1.5])
print(output)
```



You can see the creation of a neural network object does need a learning rate to be set, even though we're not using it yet. That's because our definition of the neural network class has an initialization function `__init__()` that does require it to be specified. If it's not set, the Python code will fail and throw you an error to chew on!

You can also see that the input is a list, which in Python is written inside square brackets. The output is also a list, with some numbers. Even if this

output has no real meaning because we haven't trained the network, we can be happy the thing worked with no errors.