Parameter Initialization

One of the first things that you will want do is to initialize the weight matrices and bias vectors. Their initial values for the weight matrices should not be zero since the cost reduction gradients would be the same value for each iteration (all weights associated with a particular parameters will be the same value) causing the learning algorithm to not learn. So, it's best to randomly choose the initial weight values that are between 0 and 1 and multiply them by a small scalar such as 0.01 to make the activation units active and be on the regions where activation functions' derivatives are not close to zero.

Below is a Python example that initializes the weight matrices W for each node W^i with a uniform distribution of random values. Bias vectors are initialized to zeros. It accepts an array or list which contains the dimensions for each layer and returns a matrix and bias vector for each layer.

```
def initialize_parameters(layers_dims):
    np.random.seed(1)
    parameters = {}

    # Get the number of layers
    L = len(layers_dims)

    # For each layer iniitlize the weights and bias vector
    for l in range(1, L):
        parameters["W" + str(l)] = np.random.randn(
            layers_dims[l], layers_dims[l - 1]) * 0.01
        parameters["b" + str(l)] = np.zeros((layers_dims[l], 1))

        assert parameters["W" + str(l)].shape == (
            layers_dims[l], layers_dims[l - 1])
        assert parameters["b" + str(l)].shape == (layers_dims[l], 1)

    return parameters
```