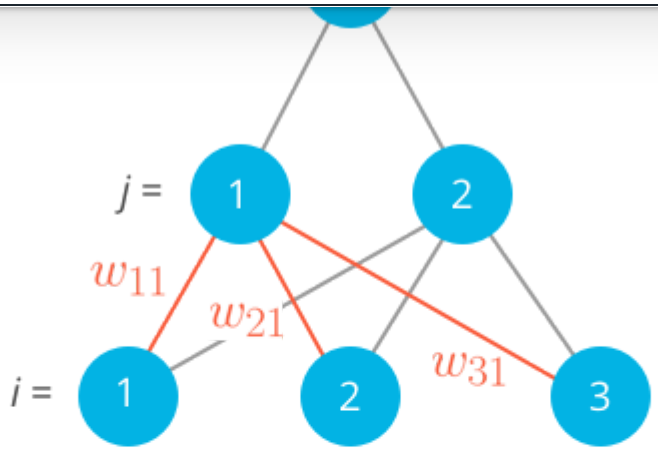

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

Calculating the input to the first hidden unit with the first column of the weights matrix.

$$h_1 = x_1 w_{11} + x_2 w_{21} + x_3 w_{31}$$

And for the second hidden layer input, you calculate the dot product of the inputs with the second column. And so on and so forth.

In NumPy, you can do this for all the inputs and all the outputs at once using `np.dot`

```
hidden_inputs = np.dot(inputs, weights_input_to_hidden)
```

You could also define your weights matrix such that it has dimensions `n_hidden` by `n_inputs` then multiply like so where the inputs form a *column vector*:

$$\begin{bmatrix} & & \end{bmatrix} \begin{bmatrix} x_1 \end{bmatrix}$$