Note: The weight indices have changed in the above image and no longer match up with the labels used in the earlier diagrams. That's because, in matrix notation, the row index always precedes the column index, so it would be misleading to label them the way we did in the neural net diagram. Just keep in mind that this is the same weight matrix as before, but rotated so the first column is now the first row, and the second column is now the second row. If we *were* to use the labels from the earlier diagram, the weights would fit into the matrix in the following locations:

$$\begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$$

Weight matrix shown with labels matching earlier diagrams.

Remember, the above is **not** a correct view of the **indices**, but it uses the labels from the earlier neural net diagrams to show you where each weight ends up in the matrix.

The important thing with matrix multiplication is that *the dimensions match*. For matrix multiplication to work, there has to be the same number of elements in the dot products. In the first example, there are three columns in the input vector, and three rows in the weights matrix. In the second example, there are three columns in the weights matrix and three rows in the input vector. If the dimensions don't match, you'll get this:

```
# Same weights and features as above, but swapped the order
hidden_inputs = np.dot(weights_input_to_hidden, features)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-11-1bfa0f615c45> in <module>()
----> 1 hidden_in = np.dot(weights_input_to_hidden, X)

ValueError: shapes (3,2) and (3,) not aligned: 2 (dim 1) != 3 (dim 0)
```

The dot product can't be computed for a 3x2 matrix and 3-element array. That's