The expression `cost = np.mean((A2 - one\_hot(Y)) \*\* 2) / 2` calculates the cost (also known as the loss) for your neural network. The cost is a measure of how well your neural network's predictions match the actual target values.

1. `A2`: This represents the output of your neural network for the current set of inputs (`X`) after the forward propagation step. It's the predicted output.

2. `one\_hot(Y)`: This is the one-hot encoded target values corresponding to the true labels `Y`. This is used to compare the predicted values (`A2`) with the actual labels.

3. `(A2 - one\_hot(Y))`: This calculates the difference between the predicted values and the actual values for each output neuron.

4. `(A2 - one\_hot(Y)) \*\* 2`: Squaring the differences element-wise. This is done to penalize larger errors more strongly than smaller errors.

5. `np.mean((A2 - one\_hot(Y)) \*\* 2)`: Taking the mean of the squared differences across all output neurons. This computes the average squared error over the entire dataset.

6. `/ 2`: Dividing the mean squared error by 2. This is a scaling factor that simplifies the derivative of the cost function during backpropagation. The choice of dividing by 2 doesn't affect the optimization process, as it only scales the cost but doesn't change the relative differences between different parameter values.

In summary, this expression calculates the mean squared error between the predicted outputs (`A2`) and the true labels (`one\_hot(Y)`) for your neural network's current iteration. This cost is used as the measure of how well the network is performing, and the goal of training is to minimize this cost by adjusting the network's parameters (weights and biases).