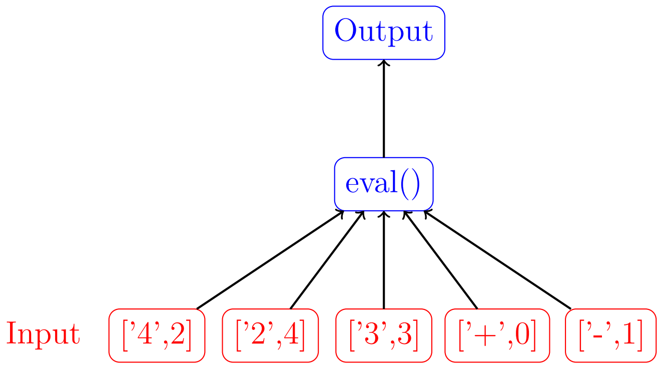
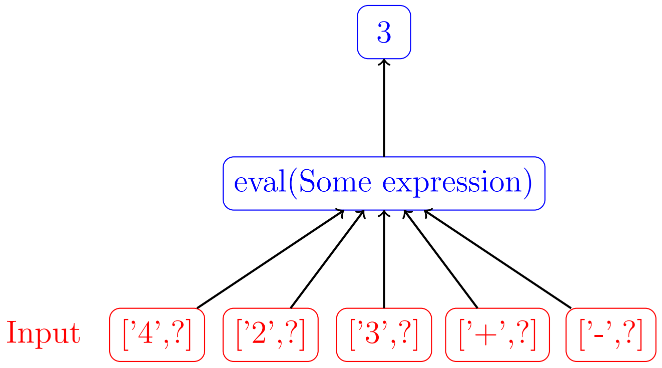
When constructing Neural Networks, we generally consider the order of inputs to neurons or the order of inputs in a layer, to be irrelivent. However, there is no reason that the order of inputs should be self-evident, and in many cases, the best order of inputs may not be known at all. To examine this, let us create a simple model for a single neuron, where it accepts any number of inputs and has a single output which is the result of applying the 'eval' function to the inputs.



Each input is a tuple where the first index is either an operator or operand, and the second index contains the input's position in the eval function. When the neuron's value is evaluated, all inputs are positioned according to their second index's value. If the tuples in the above diagram were passed into the neuron, it would result in 'eval(+ - 4 3 2)', or simply '3'.

What if we don't know the correct position for the operator and operands?
Typically, the order of inputs into a neural network isn't given much consideration, particularly when tuning the neurons to a yield a desired result. In this section, we will design a simple algorithm to tune the second index of each input tuple (the index which represents the input's position in eval()) using only the unordered inputs and a few output examples given the inputs.

Let's examine the above diagram, but eliminate all information which requires us to know the order of inputs, aside from the output:


3
eval(Some expression)
Input
['4',?]
['2',?]
['3',?]
['+',?]
['-',?]