(These are notes for me to learn more about what I'm coding)

Q Learning

The first table is the q table - the second one just has the values plugged in.
Gamma - the discount factor; a learning rate is supposed to be a coefficient representing how much to emphasize updating the q value based on the next state.
Use q learning to search for a particular circuit
We know the dimension of the target circuit - we know how many qubits exist
Same circuit only exist in a single dimension
Our actions are a universal gate set - a whole bunch of gates that we can choose from
With these gates we can get from anywhere

CNOT, H, T gates

Suppose that there is an empty circuit (with just two qubits). You could add an H or T gate to either qubit, or you could add a CNOT gate for both qubits

From the basis vectors (00 to 11), you can add a scalar multiple. As an example:

```
00 = 0.2
     0
     0
     0
```

A scalar multiple of 0.2 is applied here.

In the problem we're trying to solve, one state might be "how good would it be if we added an H gate to qubit 1 at the current state (for example, an empty circuit)?"

To build the bell circuit using Q learning, the task is to find this Q table.

There is no arbitrary limit to the number of states. A good solution might be to limit the number of gates. If we limited it to, say, 5, we would be able to identify each individual state.

Building the bell state - there are some number of total states. We assign a number to each individual state to a map to represent this.

A reward is given for a state. How to find out how good the current state is?

If the current state is the target circuit, high reward. Otherwise, we give it a small negative reward to punish it for using an additional gate.