



```
print(features)
> array([ 0.49671415, -0.1382643 ,  0.64768854])

print(features.T)
> array([ 0.49671415, -0.1382643 ,  0.64768854])

print(features[:, None])
> array([[ 0.49671415],
        [-0.1382643 ],
        [ 0.64768854]])
```

Alternatively, you can create arrays with two dimensions. Then, you can use `arr.T` to get the column vector.

```
np.array(features, ndmin=2)
> array([[ 0.49671415, -0.1382643 ,  0.64768854]])

np.array(features, ndmin=2).T
> array([[ 0.49671415],
        [-0.1382643 ],
        [ 0.64768854]])
```

I personally prefer keeping all vectors as 1D arrays, it just works better in my head.

Programming quiz

Below, you'll implement a forward pass through a 4x3x2 network, with sigmoid activation functions for both layers.

Things to do:

- Calculate the input to the hidden layer.
- Calculate the hidden layer output.
- Calculate the input to the output layer.
- Calculate the output of the network.