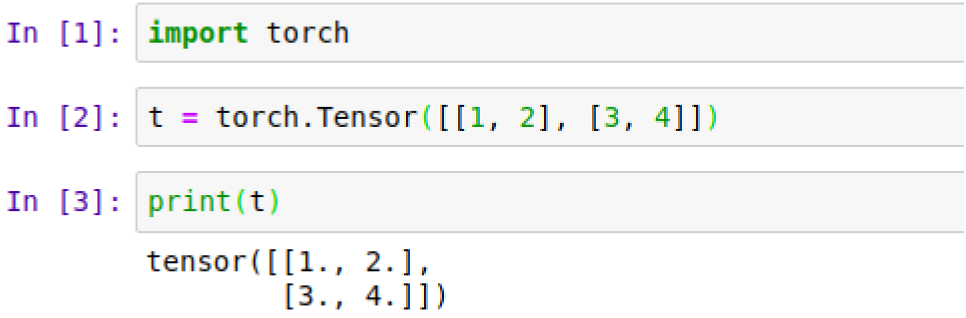
FAVORITE TORCH COMMANDS -

1. torch.tensor() :

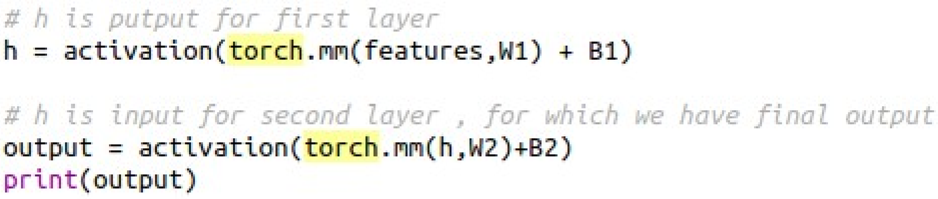
Takes parameters in the function call , to create a tensor (as an array).



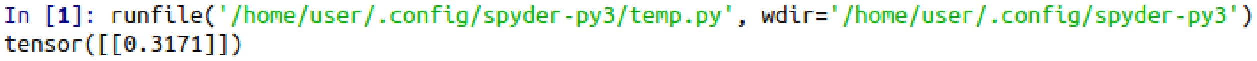
```
In [1]: import torch

In [2]: t = torch.Tensor([[1, 2], [3, 4]])

In [3]: print(t)

        tensor([[1., 2.],
                [3., 4.]])
```

2. torch.mm():

Comes in handy while caculating outputs per layer. For example -



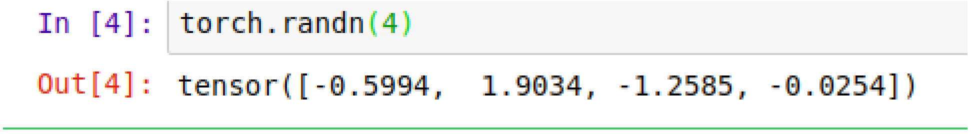
```
# h is putput for first layer
h = activation(torch.mm(features,W1) + B1)

# h is input for second layer , for which we have final output
output = activation(torch.mm(h,W2)+B2)
print(output)
```



```
In [1]: runfile('/home/user/.config/spyder-py3/temp.py', wdir='/home/user/.config/spyder-py3')
tensor([[0.3171]])
```

3. tensor.randn():

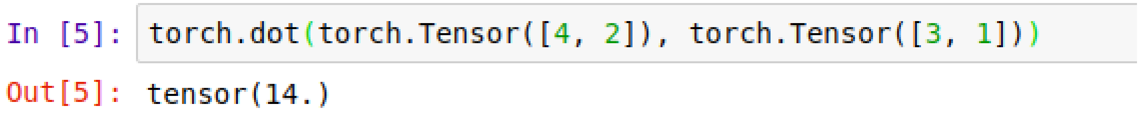
Returns a tensor filled with random numbers from a normal distribution.



```
In [4]: torch.randn(4)

Out[4]: tensor([-0.5994,  1.9034, -1.2585, -0.0254])
```

4. torch.dot():

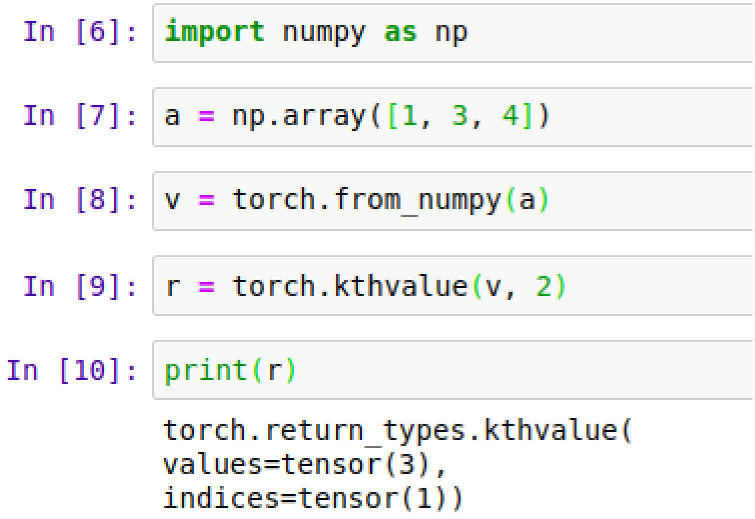
Returns the dot product of two tensors.



```
In [5]: torch.dot(torch.Tensor([4, 2]), torch.Tensor([3, 1]))

Out[5]: tensor(14.)
```

5. `torch.kthvalue():`

Returns the k-th element (start from 1) in ascending order with corresponding index.



```
In [6]: import numpy as np

In [7]: a = np.array([1, 3, 4])

In [8]: v = torch.from_numpy(a)

In [9]: r = torch.kthvalue(v, 2)

In [10]: print(r)

         torch.return_types.kthvalue(
         values=tensor(3),
         indices=tensor(1))
```