In [1]:
```
import torch
import numpy as np
```

In [14]:
```
data = [
[1,2],
[3,4]
]
x = torch.tensor(data)
print(type(x))
```

```
<class 'torch.Tensor'>
```

In [15]:
```
np_array = np.array(data)
x_np = torch.from_numpy(np_array)
print(x_np)
print(type(x_np))
```

```
tensor([[1, 2],
        [3, 4]], dtype=torch.int32)
<class 'torch.Tensor'>
```

In [22]:
```
x_ones = torch.ones_like(x)
print("One Tensor: \n",x_ones)
x_rand = torch.rand_like(x,dtype=torch.float)
print(x_rand)
```

```
One Tensor:
 tensor([[1, 1],
        [1, 1]])
tensor([[0.9368, 0.6028],
        [0.6993, 0.3815]])
```

In [17]:
```
shape = (2,3)
random_tensor = torch.rand(shape)
print(random_tensor)
print(type(random_tensor))
```

```
tensor([[0.2898, 0.6229, 0.2640],
        [0.5228, 0.0962, 0.5431]])
<class 'torch.Tensor'>
```

In [18]:
```
ones_tensor = torch.ones(shape)
print(ones_tensor)
print(type(ones_tensor))

zeros_tensor = torch.zeros(shape)
print(zeros_tensor)
print(type(zeros_tensor))
```

```
tensor([[1., 1., 1.],
        [1., 1., 1.]])
<class 'torch.Tensor'>
tensor([[0., 0., 0.],
        [0., 0., 0.]])
<class 'torch.Tensor'>
```

In [19]:
```
tensor = torch.rand(3,4)
print(tensor)
print(tensor.shape)
print(tensor.dtype)
print(tensor.device)
```

```
tensor([[0.0978, 0.5718, 0.7092, 0.2373],
        [0.5932, 0.5210, 0.3906, 0.1997],
        [0.6075, 0.5105, 0.4930, 0.8885]])
torch.Size([3, 4])
torch.float32
cpu
```

In [20]:
```
if torch.cuda.is_available():
 tensor = tensor.to('cuda')
 print("Device tensor is stored in ", tensor.device)

# Indexing, Slicing
tensor = torch.ones(4,4)
print(tensor)
print(tensor)
tensor1 = torch.zeros(4,4)
print(tensor1)
tensor2 = torch.cat([tensor,tensor1])
print(tensor2)
```

```
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
tensor([[0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]])
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]])
```

In [21]:
```
tensor.mul(tensor1)
tensor * tensor1
tensor.T
```

Out[21]:
```
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
```

In [12]:
```
tensor.add_(3)
print(tensor)
```

```
tensor([[7., 7., 7., 7.],
        [7., 7., 7., 7.],
        [7., 7., 7., 7.],
        [7., 7., 7., 7.]])
```

In [13]:
```
# from tensor to numpy
t = torch.ones(5)
print(t)
n = t.numpy()
print(n)
print(type(n))
```

```
tensor([1., 1., 1., 1., 1.])
[1. 1. 1. 1. 1.]
<class 'numpy.ndarray'>
```

In [ ]: