Manipulate PyTorch Tensors

Matrix manipulation

In [19]:

```
import torch
```

Make the matrices A and B below. Add them together to obtain a matrix C. Print these three matrices.

$$ A =\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad B =\begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} \qquad C=A+B =? $$

In [9]:

```
import numpy as np
# write your code here

A = np.array([[1,2], [3, 4]])
B = np.array([[10, 20], [30, 40]])
C = A + B

# print
print(A)
print('')
print(B)
print('')
print(C)
```

```
[[1 2]
 [3 4]]

[[10 20]
 [30 40]]

[[11 22]
 [33 44]]
```

Print the dimension, size and type of the matrix A. Remember, the commands are dim(), size() and type()

In [17]:

```
# write your code here

print(A.ndim)    # print the dimension of the matrix A
print('')
print(A.size)    # print the size of the matrix A
print('')
print(type(A))    # print the type of the matrix A
```

```
2

4

<class 'numpy.ndarray'>
```

Convert the matrix A to be an integer matrix (type LongTensor). Remember, the command is long(). Then print the type to check it was indeed converted.

In [26]:

```
# write your code here

A_long = torch.tensor(A, dtype = torch.long)

print(type(A_long))    # print the type of A_long
print('')
print(type(A))    # print the type of A
```

```
<class 'torch.Tensor'>

<class 'torch.Tensor'>

C:\Users\alber\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: To copy construct from a tensor, it is recomme
nded to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rather than torch.tensor(sourc
eTensor).
  This is separate from the ipykernel package so we can avoid doing imports until
```

Make a random 5 x 2 x 3 Tensor. The command is torch.rand. Then do the following: 1) Print the tensor, 2) Print its type, 3) Print its dimension, 4) Print its size, 5) Print the size of its middle dimension.

In [46]:

```
# write your code here

A = torch.rand(5,2,3)

print(A)
print(type(A))    # print the type of A
print(A.dim())    # print the dimension of A
print(A.size())    # print the size of A
print(A[0].dim())    # print the size of the middle (second) dimension
```

```
tensor([[[0.1661, 0.9535, 0.0349],
         [0.1373, 0.5067, 0.3667]],

        [[0.8873, 0.5244, 0.5050],
         [0.5828, 0.3709, 0.3313]],

        [[0.5031, 0.2025, 0.9241],
         [0.8626, 0.1085, 0.7283]],

        [[0.0095, 0.8186, 0.5817],
         [0.7872, 0.8827, 0.5018]],

        [[0.1271, 0.6069, 0.1658],
         [0.7814, 0.8648, 0.1189]]])
<class 'torch.Tensor'>
3
torch.Size([5, 2, 3])
2
```

Make 2 x 3 x 4 x 5 tensor filled with zeros then print it. (The command is torch.zeros). See if you can make sense of the display.

In [48]:

```
# write your code here

A = torch.zeros(2, 3, 4, 5)

print(A)
```

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

         [[0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.]],

         [[0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.]]],


        [[[0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.]],

         [[0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.]],

         [[0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.]]]])
```

In [ ]:

```

```

/