Manipulate PyTorch Tensors

Matrix manipulation

```
import torch
```

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

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

```

# write your code here

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

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

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

tensor([[10, 20],
        [30, 40]])

tensor([[11, 22],
        [33, 44]])
```

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

```

# write your code here

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

```
2

torch.Size([2, 2])

torch.LongTensor
```

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.

```

# write your code here

```

```
A_long = A.long()

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

```
torch.LongTensor

torch.LongTensor
```

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.

```

# write your code here

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

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

```
tensor([[[0.7194, 0.2755, 0.8957],
         [0.4157, 0.7653, 0.5056]],

        [[0.2037, 0.6285, 0.4734],
         [0.6616, 0.5013, 0.8263]],

        [[0.8506, 0.6952, 0.9736],
         [0.6165, 0.1984, 0.2900]],

        [[0.4928, 0.7679, 0.3038],
         [0.6613, 0.5541, 0.0488]],

        [[0.3075, 0.0140, 0.7910],
         [0.1384, 0.1092, 0.9841]]])
torch.FloatTensor
3
torch.Size([5, 2, 3])
```

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.

```

# 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.]]]])
```