Introduction

The objective of this session is to practice with basic tensor manipulations in pytorch, to understand the relation between a tensor and its underlying storage, and get a sense of the efficiency of tensor-based computation compared to their equivalent python iterative implementations.

1. **Multiple views of a storage**

   Generate the matrix

1	2	1	1	1	1	2	1	1	1	1	2	1
2	2	2	2	2	2	2	2	2	2	2	2	2
1	2	1	1	1	1	2	1	1	1	1	2	1
1	2	1	3	3	1	2	1	3	3	1	2	1
1	2	1	3	3	1	2	1	3	3	1	2	1
1	2	1	1	1	1	2	1	1	1	1	2	1
2	2	2	2	2	2	2	2	2	2	2	2	2
1	2	1	1	1	1	2	1	1	1	1	2	1
1	2	1	3	3	1	2	1	3	3	1	2	1
1	2	1	3	3	1	2	1	3	3	1	2	1
1	2	1	1	1	1	2	1	1	1	1	2	1
2	2	2	2	2	2	2	2	2	2	2	2	2
1	2	1	1	1	1	2	1	1	1	1	2	1

with no python loop.
Hint: Use torch.full, and the slicing operator.

2. **Flops per second**

Generate two square matrices of dimension 5000 × 5000 filled with random Gaussian coefficients, compute their product, measure the time it takes, and estimate how many floating point products have been executed per second (should be in the billions or tens of billions).

Hint: Use torch.empty, torch.normal , torch.mm, and time.perf counter.

3. **Playing with strides**

Write a function mul row, using python loops (and not even slicing operators), that gets a 2d tensor as argument, and returns a tensor of same size, equal to the one given as argument, with the first row kept unchanged, the second multiplied by two, the third by three, etc.
For instance:

- m = torch.full((4, 8), 2.0)
- m

```
tensor([[2., 2., 2., 2., 2., 2., 2., 2.],
        [2., 2., 2., 2., 2., 2., 2., 2.],
        [2., 2., 2., 2., 2., 2., 2., 2.],
        [2., 2., 2., 2., 2., 2., 2., 2.]])
>>> mul_row(m)
tensor([[2., 2., 2., 2., 2., 2., 2., 2.],
        [4., 4., 4., 4., 4., 4., 4., 4.],
        [6., 6., 6., 6., 6., 6., 6., 6.],
        [8., 8., 8., 8., 8., 8., 8., 8.]])
```

Then, write a second version named mul_row_fast, using tensor operations.
Apply both versions to a matrix of size 1, 000 × 400 and measure the time each takes (there should be more than two orders of magnitude difference).

Hint: Use broadcasting and torch.arange, torch.view, torch.mul, and time.perf counter.