1. Tensor

> A Tensor is the core data structure and computational unit of PyTorch.

> PyTorch tensors are multi-dimensional arrays that contain values of the same data type. They can be thought of as a NumPy array, but highly optimized and GPU-compatible.

> A Tensor is created from a Python list as follows:

Summary

No.	Code	Tags	Description
1	`torch.tensor(list)`	Python, PyTorch, (tutorial, how-to), (tensor, initialization, (from, list))	Initialize a PyTorch tensor from a Python list.
2	`torch.arange(start: int, end: int, step: int)`	Python, PyTorch, (tutorial, how-to), (tensor, initialization, (from, range))	Initialize a PyTorch tensor from a Python range.
3	`torch.empty(rows: int, columns: int)`	Python, PyTorch, (tutorial, how-to), (tensor, initialization, (with, null, as, not, values))	Initialize an empty tensor of the specified shape.
4	`torch.zeros(rows: int, columns: int)`	Python, PyTorch, (tutorial, how-to), (tensor, initialization, (with, zeros))	Like `empty`, but initializing with 0s instead of null values.
5	`torch.ones(rows: int, columns: int)`	Python, PyTorch, (tutorial, how-to), (tensor, initialization, (with, multiple, one))	Like `empty`, but initializing with 1s instead.
6	`torch.rand(rows: int, columns: int)`	Python, PyTorch, (tutorial, how-to), (tensor, initialization, (with, random, float, numbers, (from, statistics, probability, distribution, uniform, distribution)))	Like `empty`, but initializing with random decimal numbers sampled from a **uniform distribution** instead.
7	`torch.rand(rows: int, columns: int)`	Python, PyTorch, (tutorial, how-to), (tensor, initialization, (with, random, float, numbers, (from, statistics, probability, distribution, normal distribution)))	Like `empty`, but initializing with random decimal numbers sampled from a **normal distribution** instead.

Notes

In [51]:

```
import torch

# Create tensor from a list
x = torch.tensor([1, 2, 3])

print(x)
#tensor([1, 2, 3])
```

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

> An uninitialized Tensor is created using torch.empty(shape)

In [52]:

```
# Create a 3x2 tensor with uninitialized values
x = torch.empty(3, 2)
```

> Tensors initialized with zeros or ones are created using torch.zeros(shape) or torch.ones(shape)

In [53]:

```
# Create a 3x2 tensor with zeros
x = torch.zeros(3, 2)

"""
tensor([[0., 0.],
        [0., 0.],
        [0., 0.]])
"""

print(x)
```

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

In [54]:

```
# Create a 3x2 tensor with ones
x = torch.ones(3, 2)

"""
tensor([[1., 1.],
        [1., 1.],
        [1., 1.]])

"""

print(x)
```

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

> Tensors initialized with random values from a uniform distribution on the interval of 0 to 1 (not including 1) are created using torch.rand(shape)

In [55]:

```
# Create a 2x2 tensor with random values from the uniform distribution
x = torch.rand(2, 2)

"""
tensor([[0.6051, 0.0569],
        [0.7959, 0.0452]])
"""

print(x)
```

```
tensor([[0.1914, 0.8260],
        [0.8402, 0.0820]])
```

> Tensors initialized with random numbers from a normal distribution with mean 0 and variance 1 are created using torch.randn(shape)n

In [56]:

```
# Create a 2x2 tensor with random values from the standard normal distributi
x = torch.randn(2, 2)

"""
tensor([[ 0.7346, -0.3198],
        [ 0.9044,  0.0995]])
"""

print(x)
```

```
tensor([[ 2.0006, -1.1012],
        [ 0.4398,  2.0937]])
```

> `torch.arange(start, end, step)` is used to create a 1-D tensor with evenly spaced values within a given interval

In [57]:

```
# Create a 1-D tensor
x = torch.arange(0, 10, 2)

print(x)
# tensor([0, 2, 4, 6, 8])
```

```
tensor([0, 2, 4, 6, 8])
```