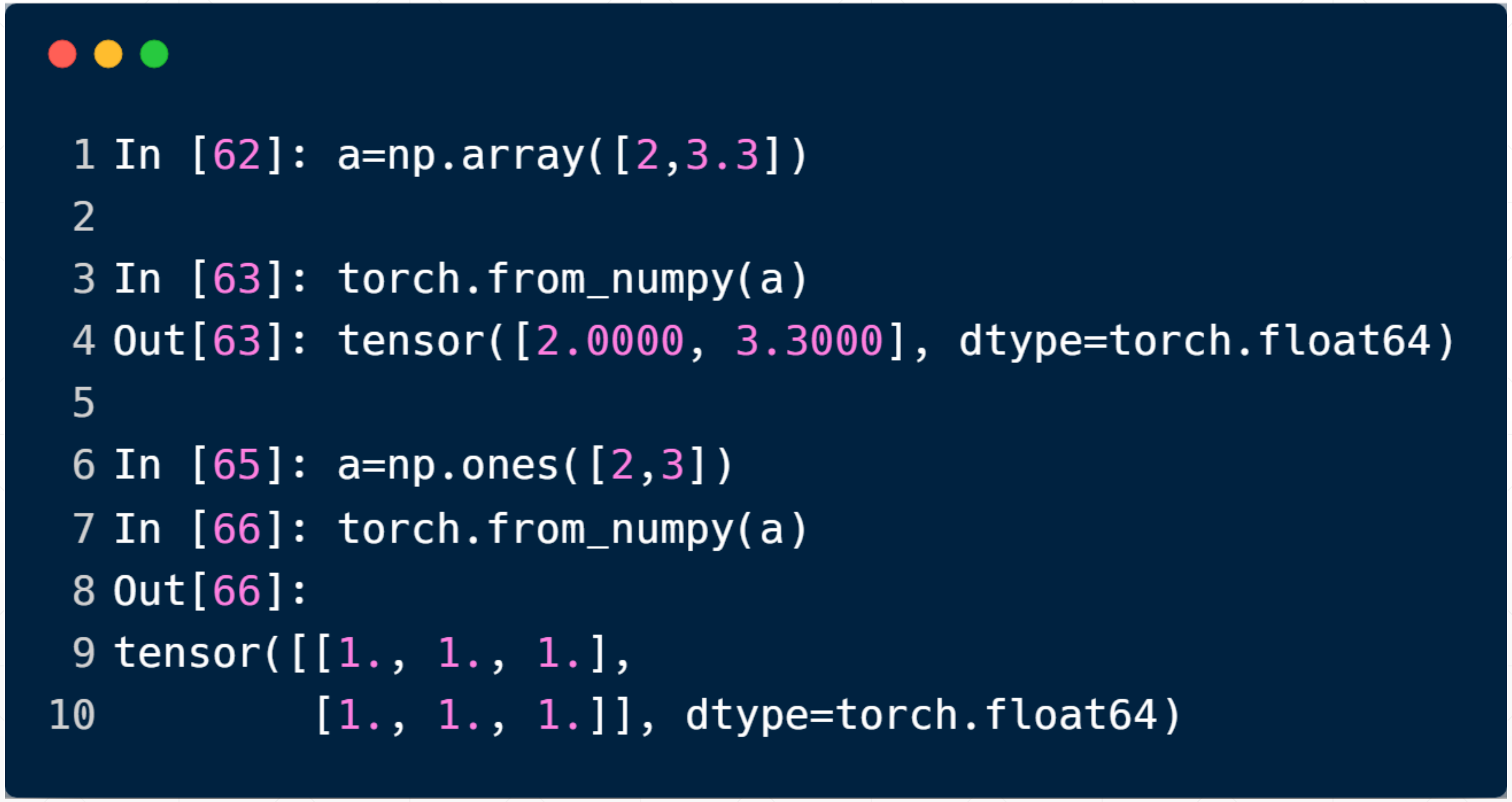
PyTorch

创建Tensor

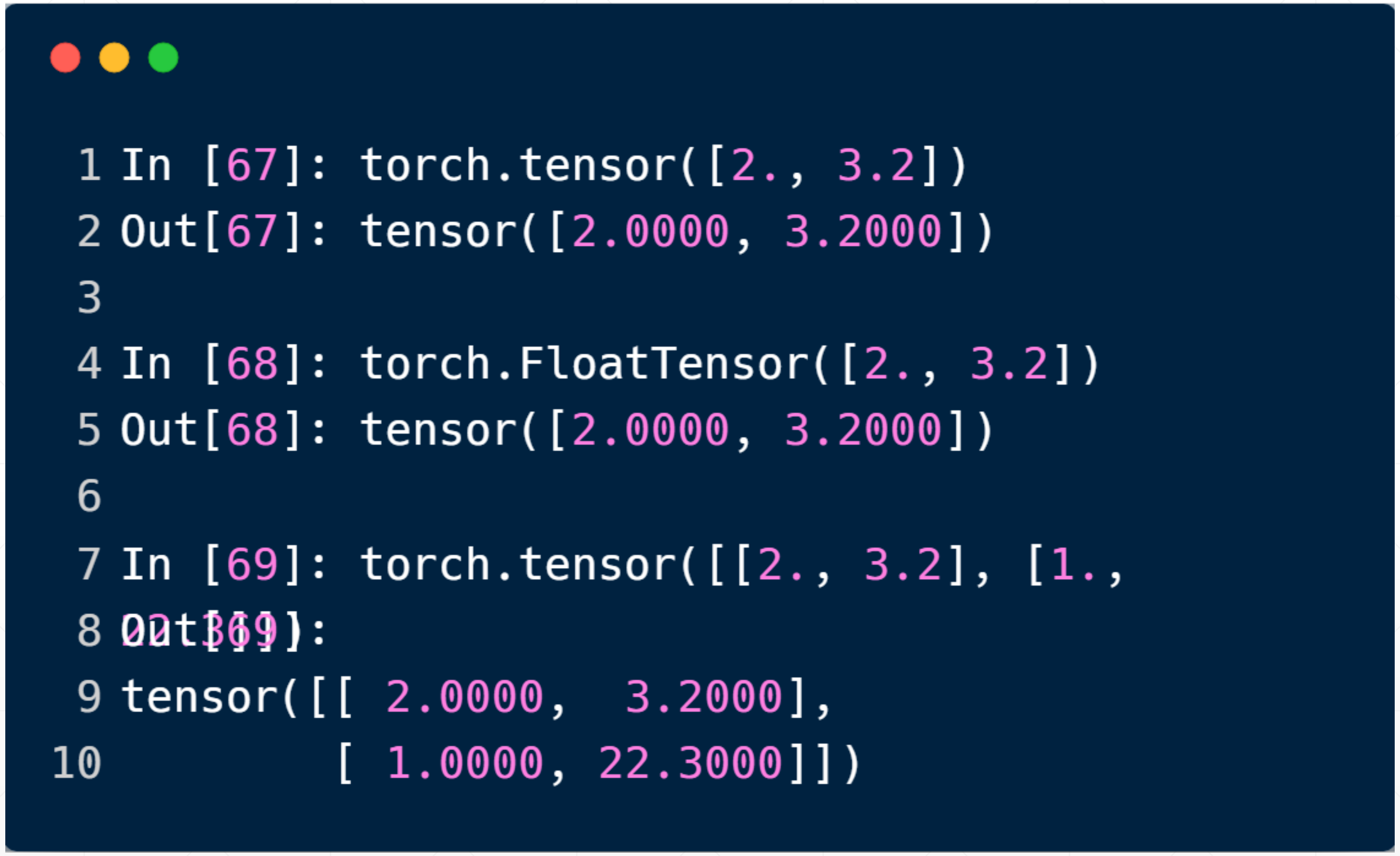
主讲人：龙良曲

Import from numpy



```
In [62]: a=np.array([2,3.3])

In [63]: torch.from_numpy(a)
Out[63]: tensor([2.0000, 3.3000], dtype=torch.float64)

In [65]: a=np.ones([2,3])
In [66]: torch.from_numpy(a)
Out[66]:
tensor([[1., 1., 1.],
        [1., 1., 1.]], dtype=torch.float64)
```

Import from List



```
In [67]: torch.tensor([2., 3.2])
Out[67]: tensor([2.0000, 3.2000])

In [68]: torch.FloatTensor([2., 3.2])
Out[68]: tensor([2.0000, 3.2000])

In [69]: torch.tensor([[2., 3.2], [1.,
22.3]])
Out[69]:
tensor([[ 2.0000,  3.2000],
        [ 1.0000, 22.3000]])
```

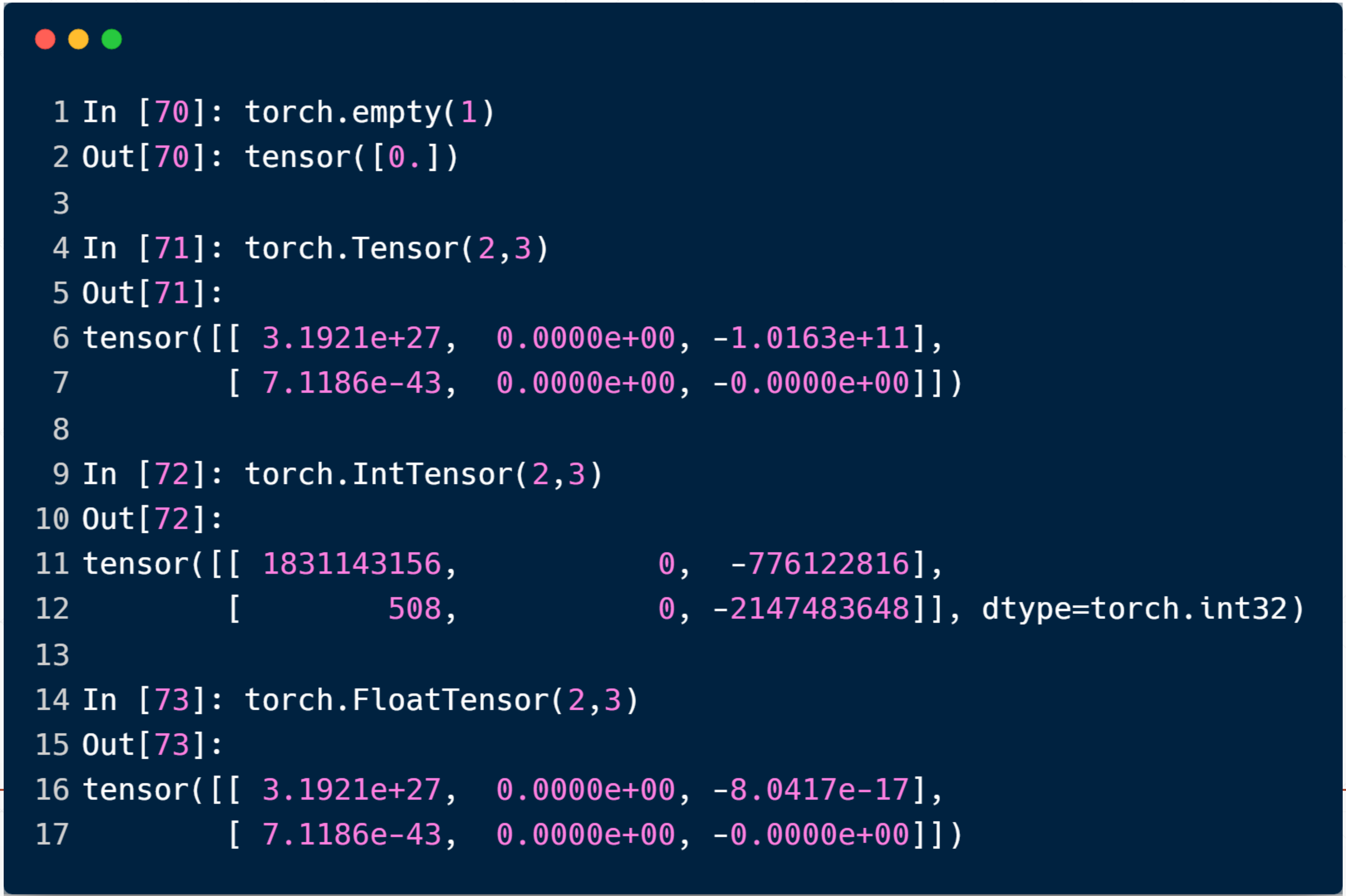
uninitialized

- Torch.empty()

- Torch.FloatTensor(d1, d2, d3)
  - NOT torch.FloatTensor([1, 2]) = torch.tensor([1, 2])

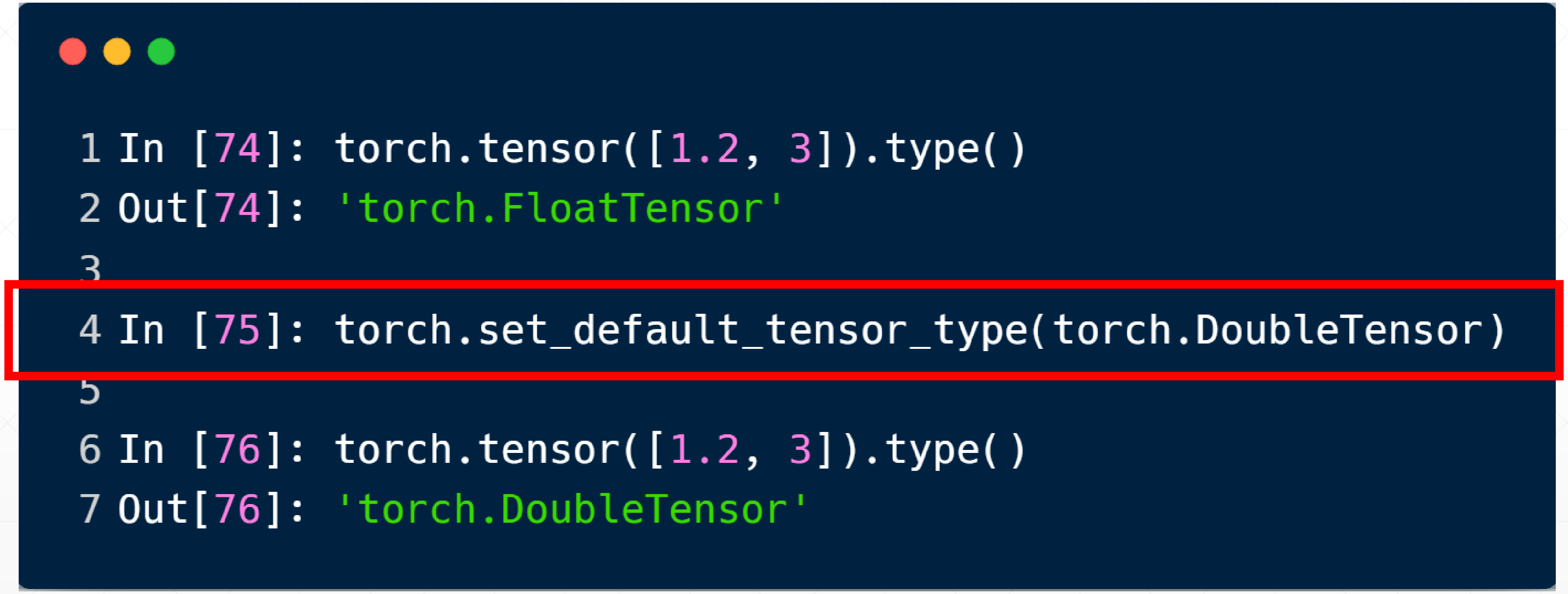
- Torch.IntTensr(d1, d2, d3)

uninitialized



```
In [70]: torch.empty(1)
Out[70]: tensor([0.])

In [71]: torch.Tensor(2,3)
Out[71]:
tensor([[ 3.1921e+27,  0.0000e+00, -1.0163e+11],
        [ 7.1186e-43,  0.0000e+00, -0.0000e+00]])

In [72]: torch.IntTensor(2,3)
Out[72]:
tensor([[ 1831143156,           0,  -776122816],
        [        508,           0, -2147483648]], dtype=torch.int32)

In [73]: torch.FloatTensor(2,3)
Out[73]:
tensor([[ 3.1921e+27,  0.0000e+00, -8.0417e-17],
        [ 7.1186e-43,  0.0000e+00, -0.0000e+00]])
```

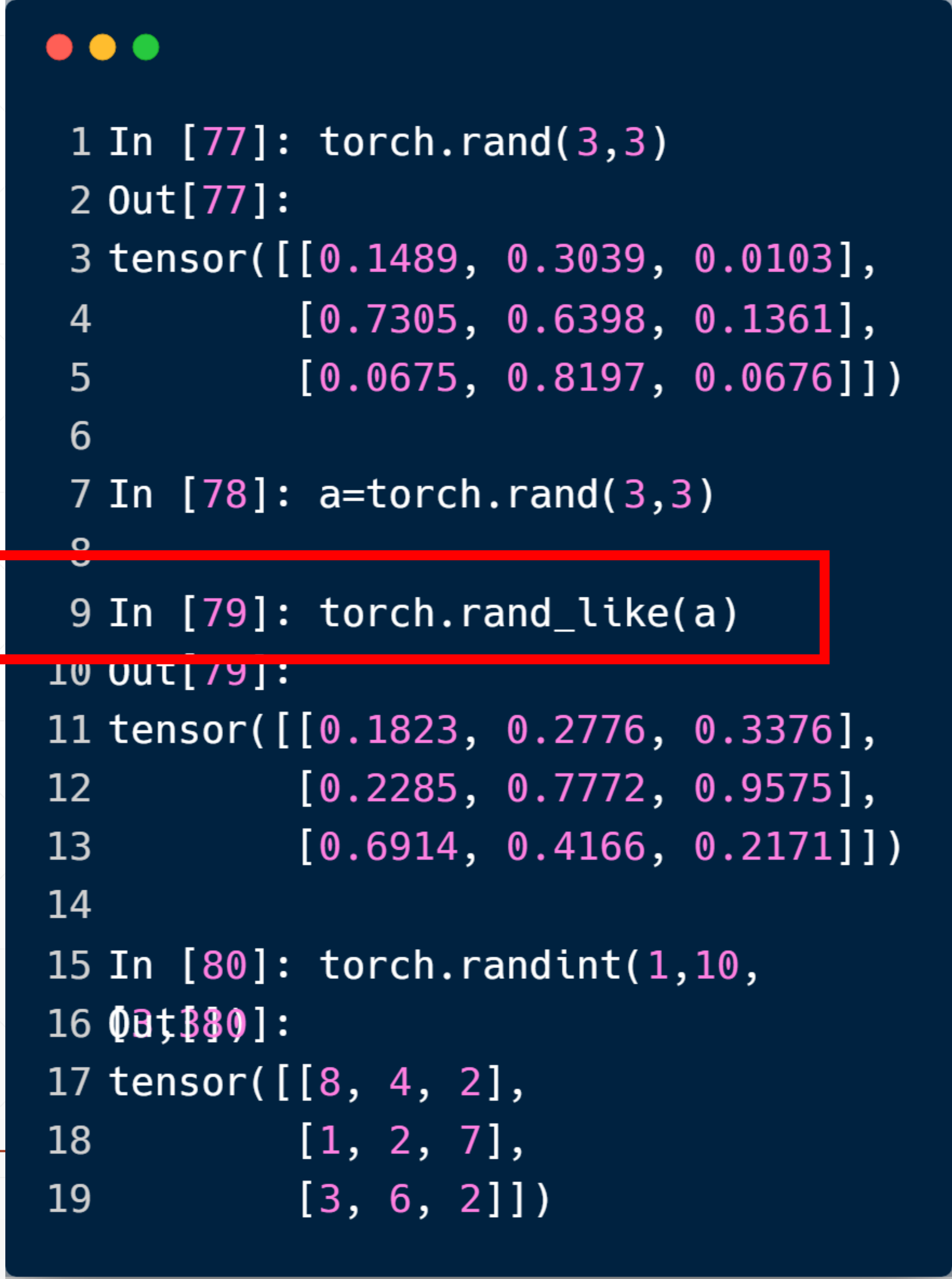
set default type



```
In [74]: torch.tensor([1.2, 3]).type()
Out[74]: 'torch.FloatTensor'

In [75]: torch.set_default_tensor_type(torch.DoubleTensor)

In [76]: torch.tensor([1.2, 3]).type()
Out[76]: 'torch.DoubleTensor'
```

rand/rand_like, randint

- [0, 1]
- [min, max)
- *_like



```
In [77]: torch.rand(3,3)
Out[77]:
tensor([[0.1489, 0.3039, 0.0103],
        [0.7305, 0.6398, 0.1361],
        [0.0675, 0.8197, 0.0676]])

In [78]: a=torch.rand(3,3)


In [79]: torch.rand_like(a)
Out[79]:
tensor([[0.1823, 0.2776, 0.3376],
        [0.2285, 0.7772, 0.9575],
        [0.6914, 0.4166, 0.2171]])

In [80]: torch.randint(1,10,
(3,3))
Out[80]:
tensor([[8, 4, 2],
        [1, 2, 7],
        [3, 6, 2]])
```

randn

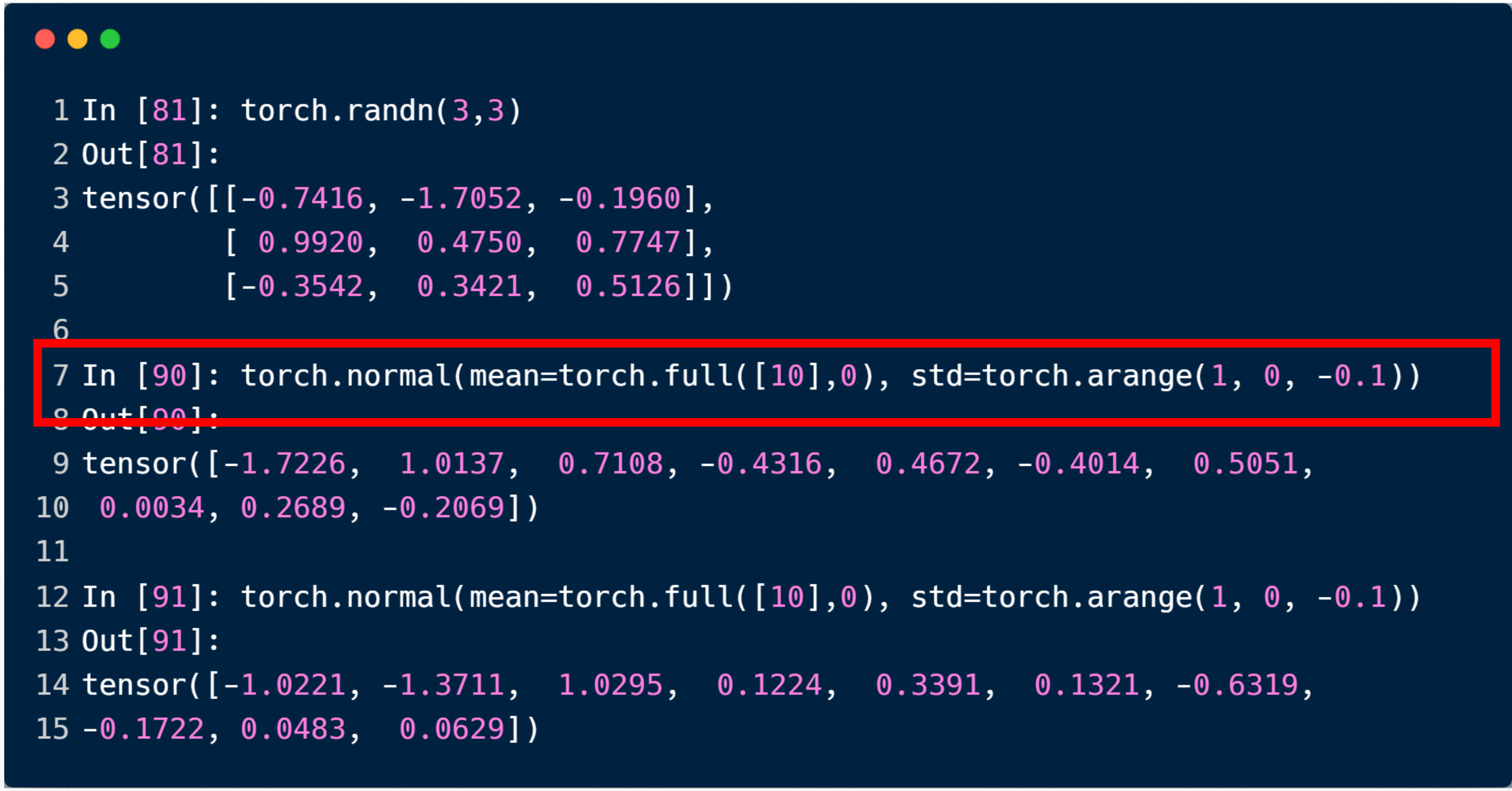
```
>>> torch.normal(mean=torch.arange(1., 11.), std=torch.arange(1, 0, -0.1))
tensor([  1.0425,   3.5672,   2.7969,   4.2925,   4.7229,   6.2134,
          8.0505,   8.1408,   9.0563,  10.0566])
```

- N(0, 1)
- N(u, std)

可以每个数据，单独按照这个位置给定的均值和标准差随机，所以mean和std两个参数，都是tensor

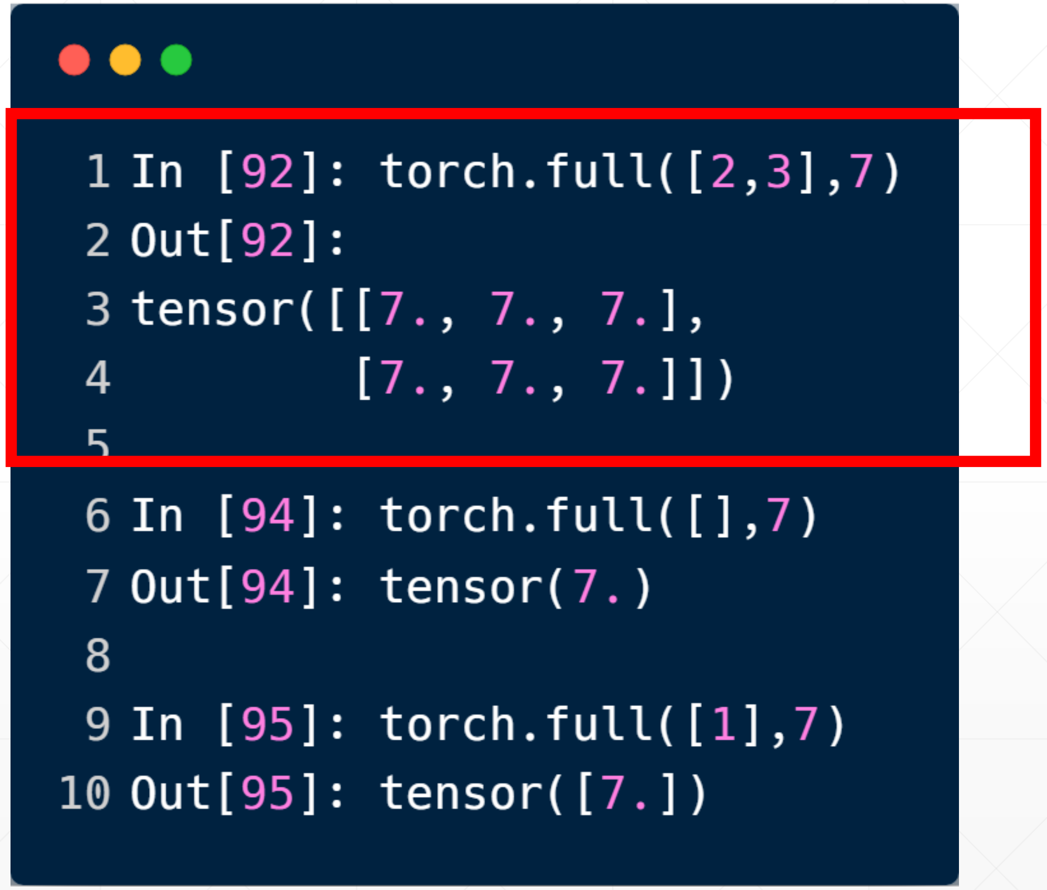
mean (Tensor) – the tensor of per-element means

std (Tensor) – the tensor of per-element standard deviations



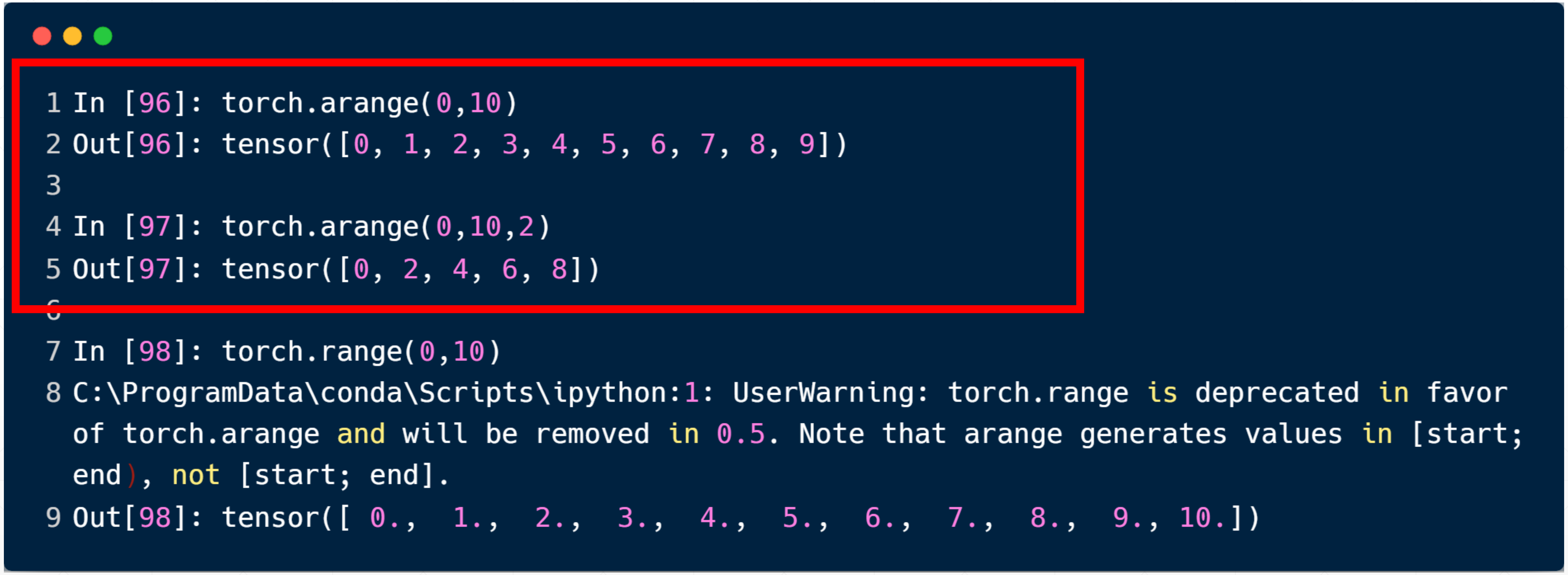
```
In [81]: torch.randn(3,3)
Out[81]:
tensor([[-0.7416, -1.7052, -0.1960],
        [ 0.9920,  0.4750,  0.7747],
        [-0.3542,  0.3421,  0.5126]])

In [90]: torch.normal(mean=torch.full([10],0), std=torch.arange(1, 0, -0.1))
Out[90]:
tensor([-1.7226,  1.0137,  0.7108, -0.4316,  0.4672, -0.4014,  0.5051,
 0.0034, 0.2689, -0.2069])

In [91]: torch.normal(mean=torch.full([10],0), std=torch.arange(1, 0, -0.1))
Out[91]:
tensor([-1.0221, -1.3711,  1.0295,  0.1224,  0.3391,  0.1321, -0.6319,
-0.1722, 0.0483,  0.0629])
```

full



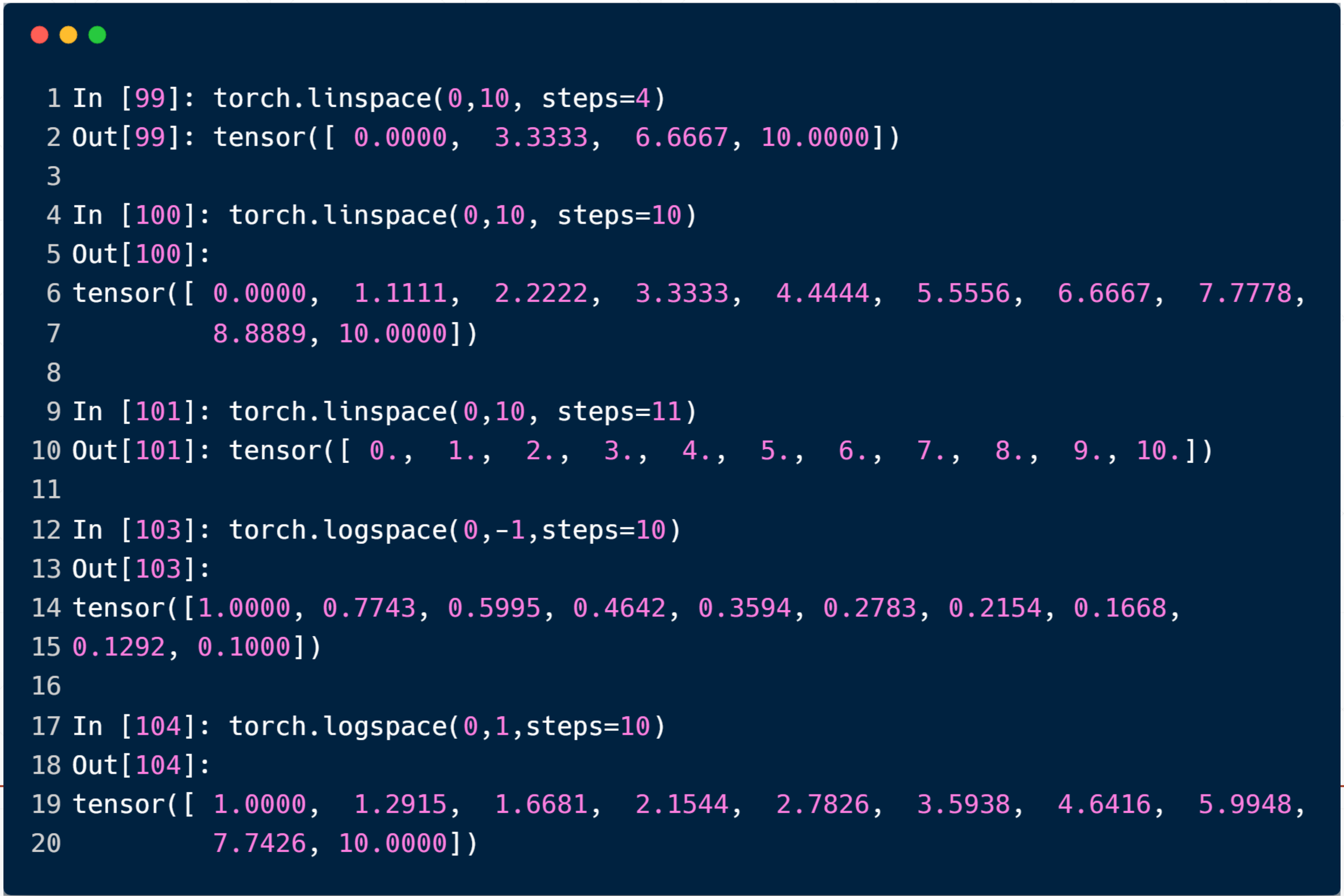
```
In [92]: torch.full([2,3],7)
Out[92]:
tensor([[7., 7., 7.],
        [7., 7., 7.]])

In [94]: torch.full([],7)
Out[94]: tensor(7.)

In [95]: torch.full([1],7)
Out[95]: tensor([7.])
```

arange/range



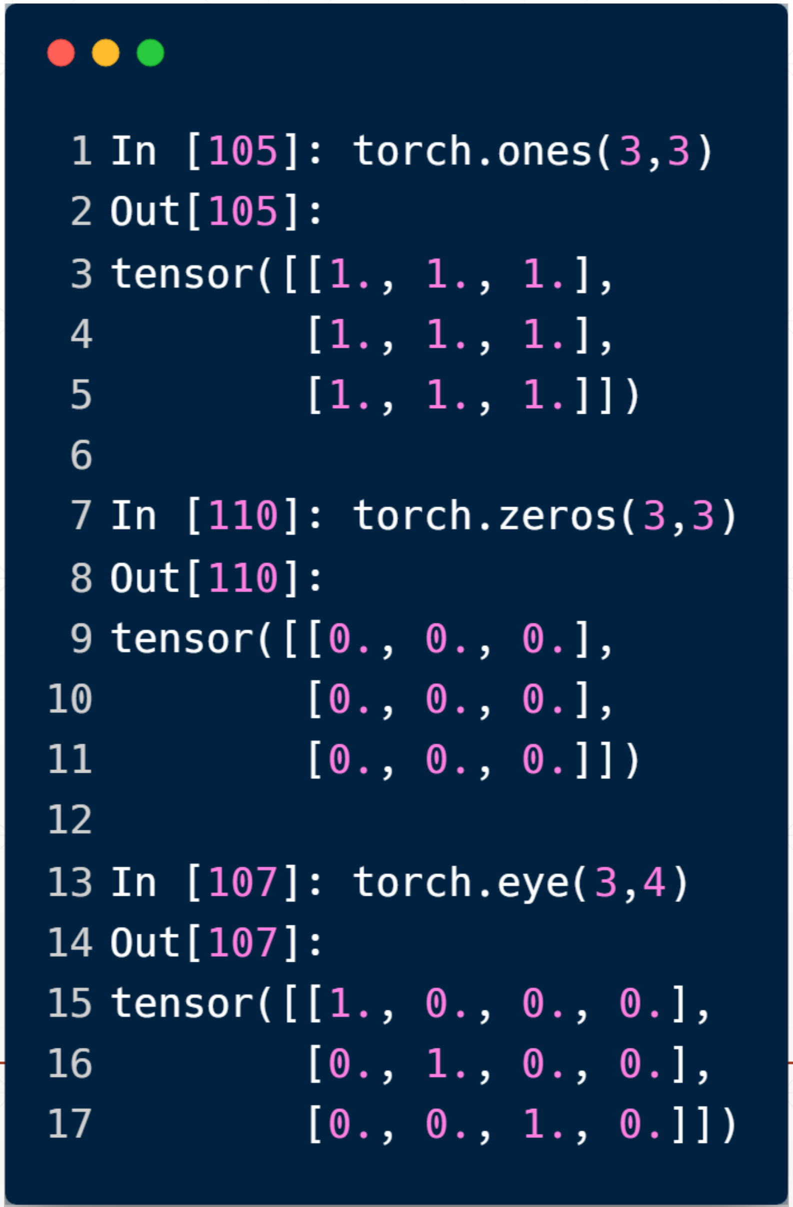
```
In [96]: torch.arange(0,10)
Out[96]: tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [97]: torch.arange(0,10,2)
Out[97]: tensor([0, 2, 4, 6, 8])

In [98]: torch.range(0,10)
C:\ProgramData\conda\Scripts\ipython:1: UserWarning: torch.range is deprecated in favor
  of torch.arange and will be removed in 0.5. Note that arange generates values in [start;
  end), not [start; end].
Out[98]: tensor([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

linspace/logspace



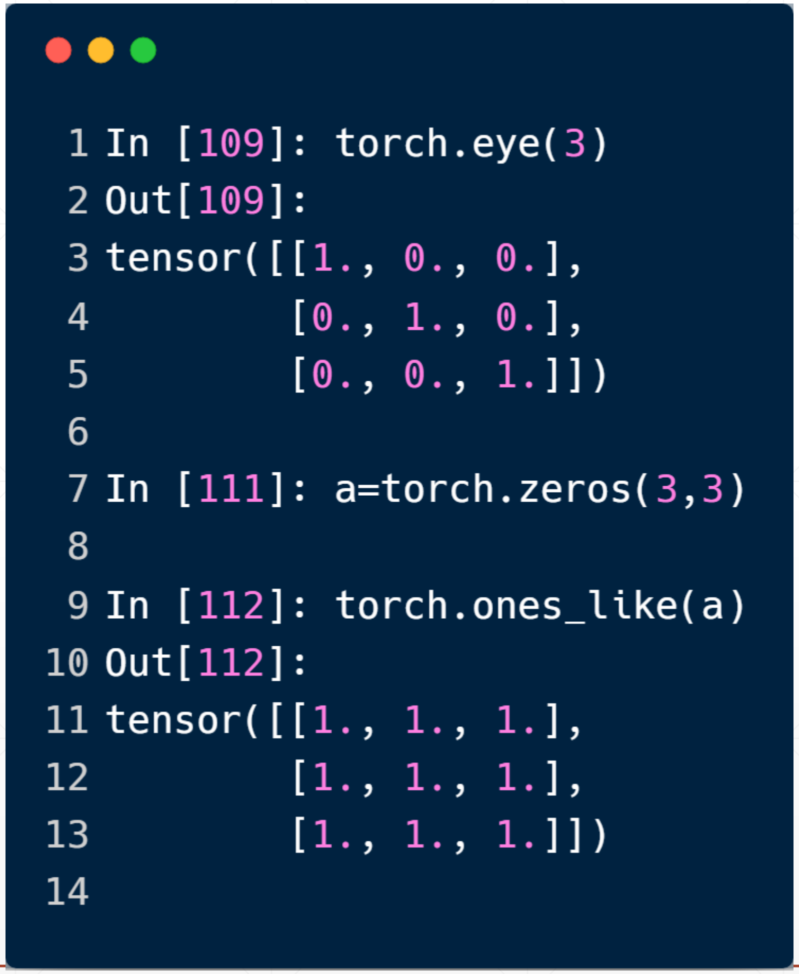
```
In [99]: torch.linspace(0,10, steps=4)
Out[99]: tensor([ 0.0000,  3.3333,  6.6667, 10.0000])

In [100]: torch.linspace(0,10, steps=10)
Out[100]:
tensor([ 0.0000,  1.1111,  2.2222,  3.3333,  4.4444,  5.5556,  6.6667,  7.7778,
         8.8889, 10.0000])

In [101]: torch.linspace(0,10, steps=11)
Out[101]: tensor([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])

In [103]: torch.logspace(0,-1,steps=10)
Out[103]:
tensor([1.0000, 0.7743, 0.5995, 0.4642, 0.3594, 0.2783, 0.2154, 0.1668,
0.1292, 0.1000])

In [104]: torch.logspace(0,1,steps=10)
Out[104]:
tensor([ 1.0000,  1.2915,  1.6681,  2.1544,  2.7826,  3.5938,  4.6416,  5.9948,
         7.7426, 10.0000])
```

Ones/zeros/eye



```
In [105]: torch.ones(3,3)
Out[105]:
tensor([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]])

In [110]: torch.zeros(3,3)
Out[110]:
tensor([[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]])

In [107]: torch.eye(3,4)
Out[107]:
tensor([[1., 0., 0., 0.],
        [0., 1., 0., 0.],
        [0., 0., 1., 0.]])
```

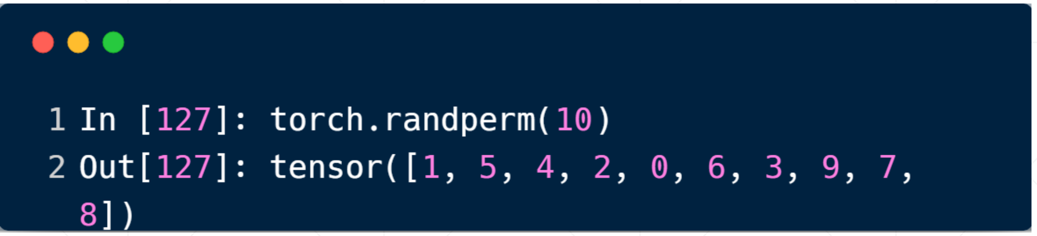
Ones/zeros/eye



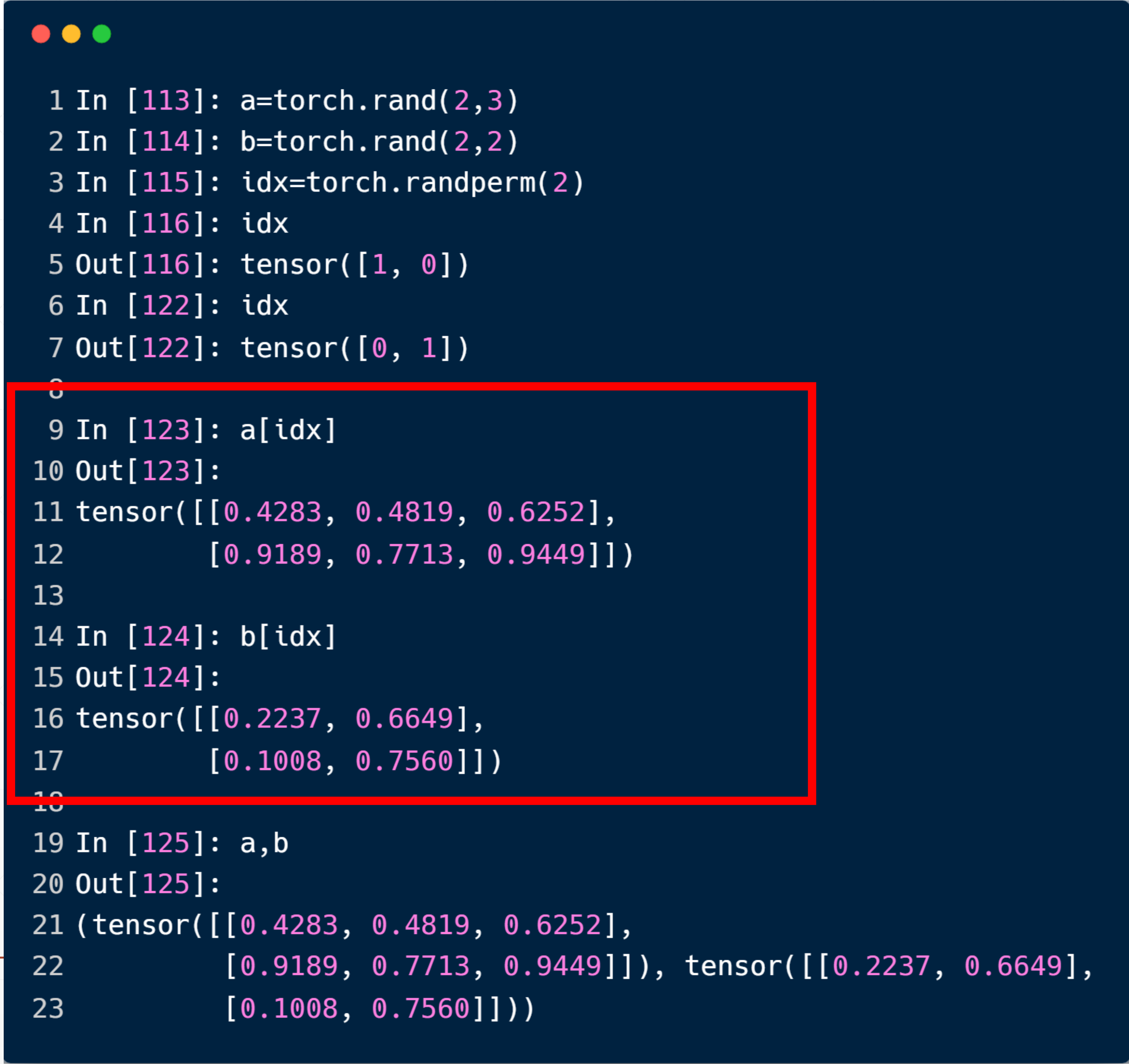
```
In [109]: torch.eye(3)
Out[109]:
tensor([[1., 0., 0.],
        [0., 1., 0.],
        [0., 0., 1.]])

In [111]: a=torch.zeros(3,3)

In [112]: torch.ones_like(a)
Out[112]:
tensor([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]])

```

randperm

- random.shuffle



```
In [127]: torch.randperm(10)
Out[127]: tensor([1, 5, 4, 2, 0, 6, 3, 9, 7,
 8])
```



```
In [113]: a=torch.rand(2,3)
In [114]: b=torch.rand(2,2)
In [115]: idx=torch.randperm(2)
In [116]: idx
Out[116]: tensor([1, 0])
In [122]: idx
Out[122]: tensor([0, 1])

In [123]: a[idx]
Out[123]:
tensor([[0.4283, 0.4819, 0.6252],
        [0.9189, 0.7713, 0.9449]])

In [124]: b[idx]
Out[124]:
tensor([[0.2237, 0.6649],
        [0.1008, 0.7560]])

In [125]: a,b
Out[125]:
(tensor([[0.4283, 0.4819, 0.6252],
         [0.9189, 0.7713, 0.9449]]), tensor([[0.2237, 0.6649],
         [0.1008, 0.7560]]))
```

下一课时

Tensor切片

Thank You.