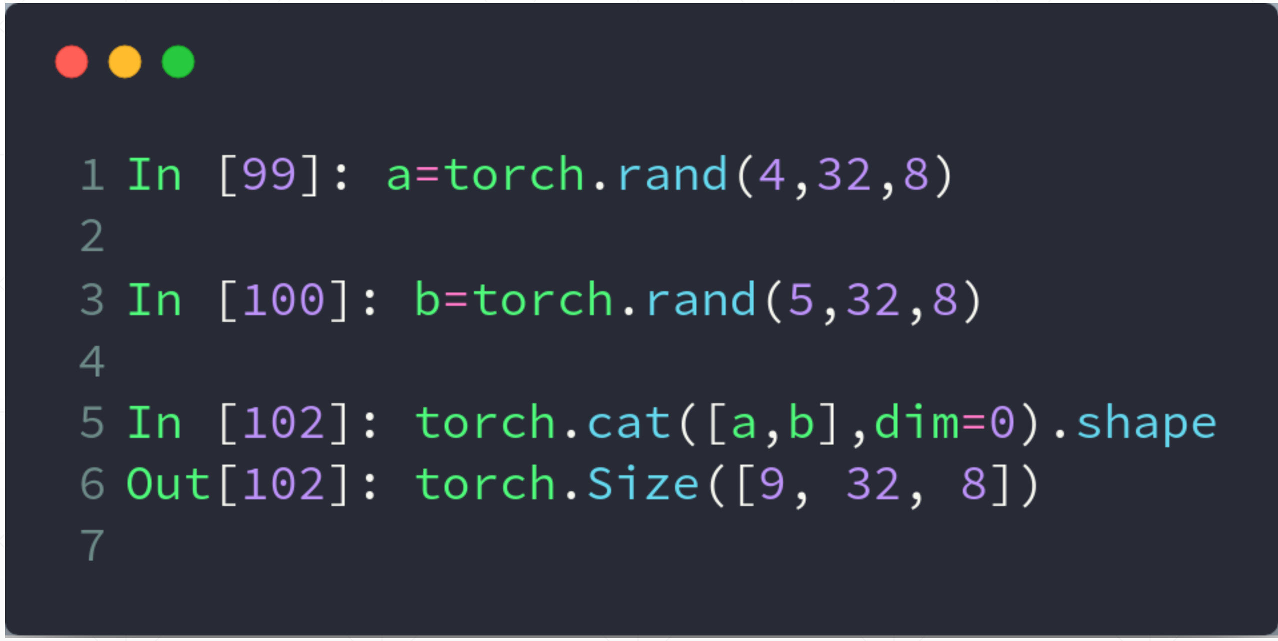
Merge or split

- Cat
- Stack
- Split
- Chunk

https://blog.openai.com/generative-models/

cat

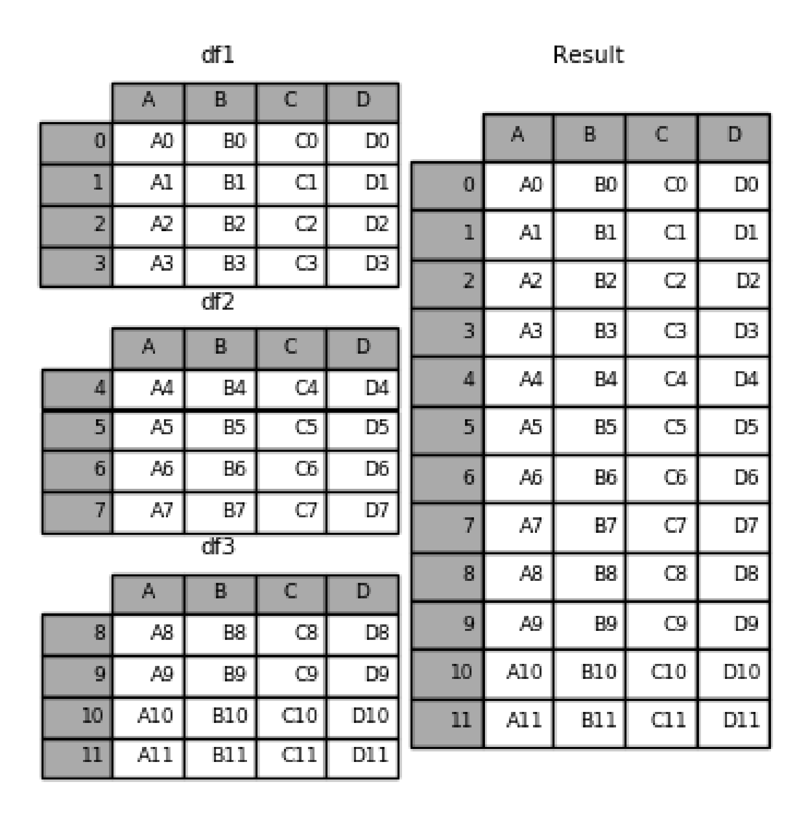
- Statistics about scores
  - [class1-4, students, scores]
  - [class5-9, students, scores]



```
In [99]: a=torch.rand(4,32,8)

In [100]: b=torch.rand(5,32,8)

In [102]: torch.cat([a,b],dim=0).shape
Out[102]: torch.Size([9, 32, 8])
```

Along distinct dim/axis

- *Dim=d*



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

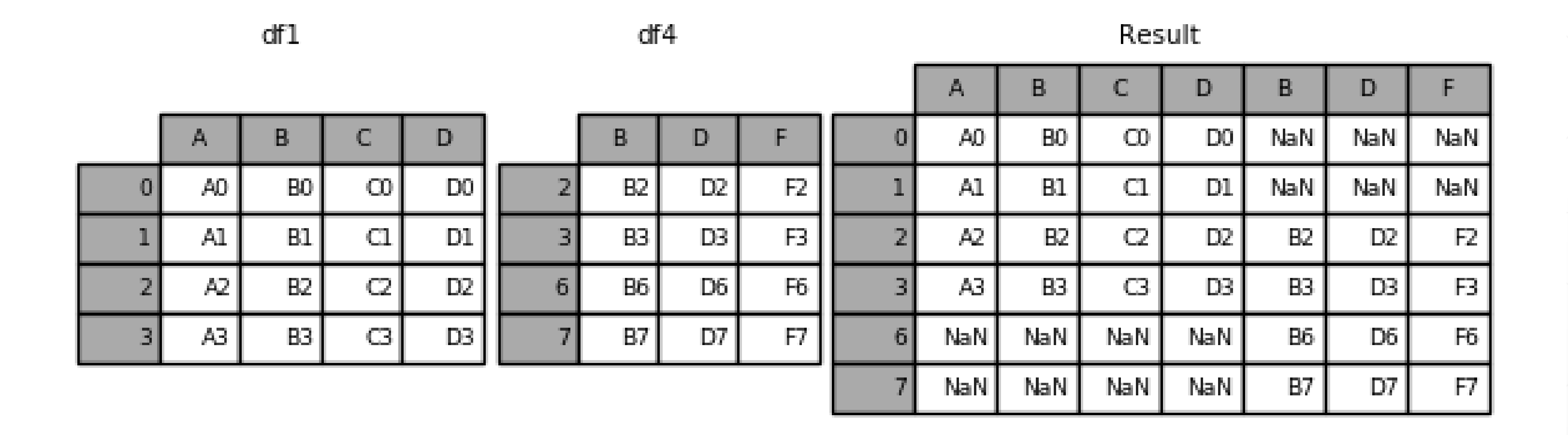
	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Result

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

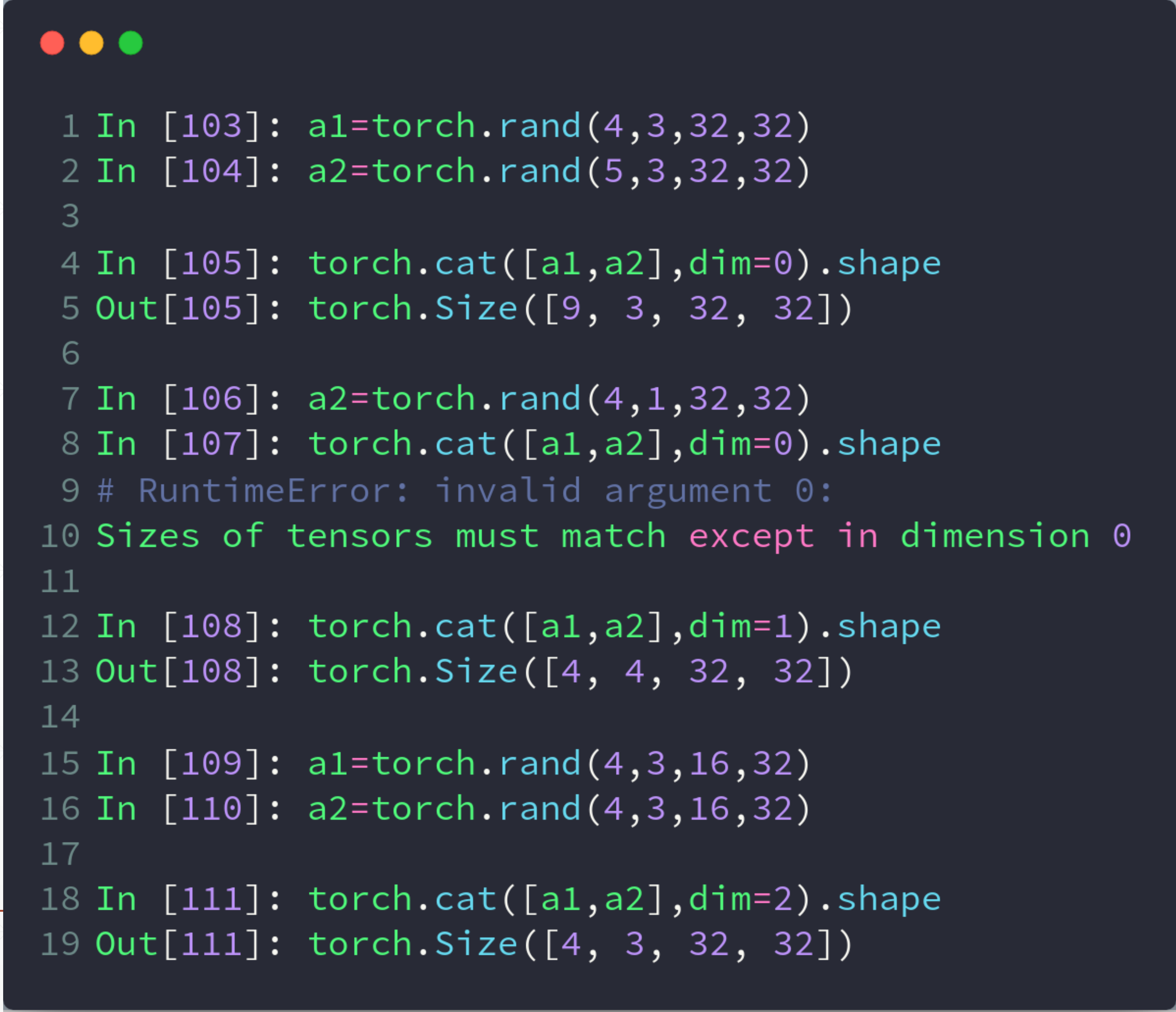
df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

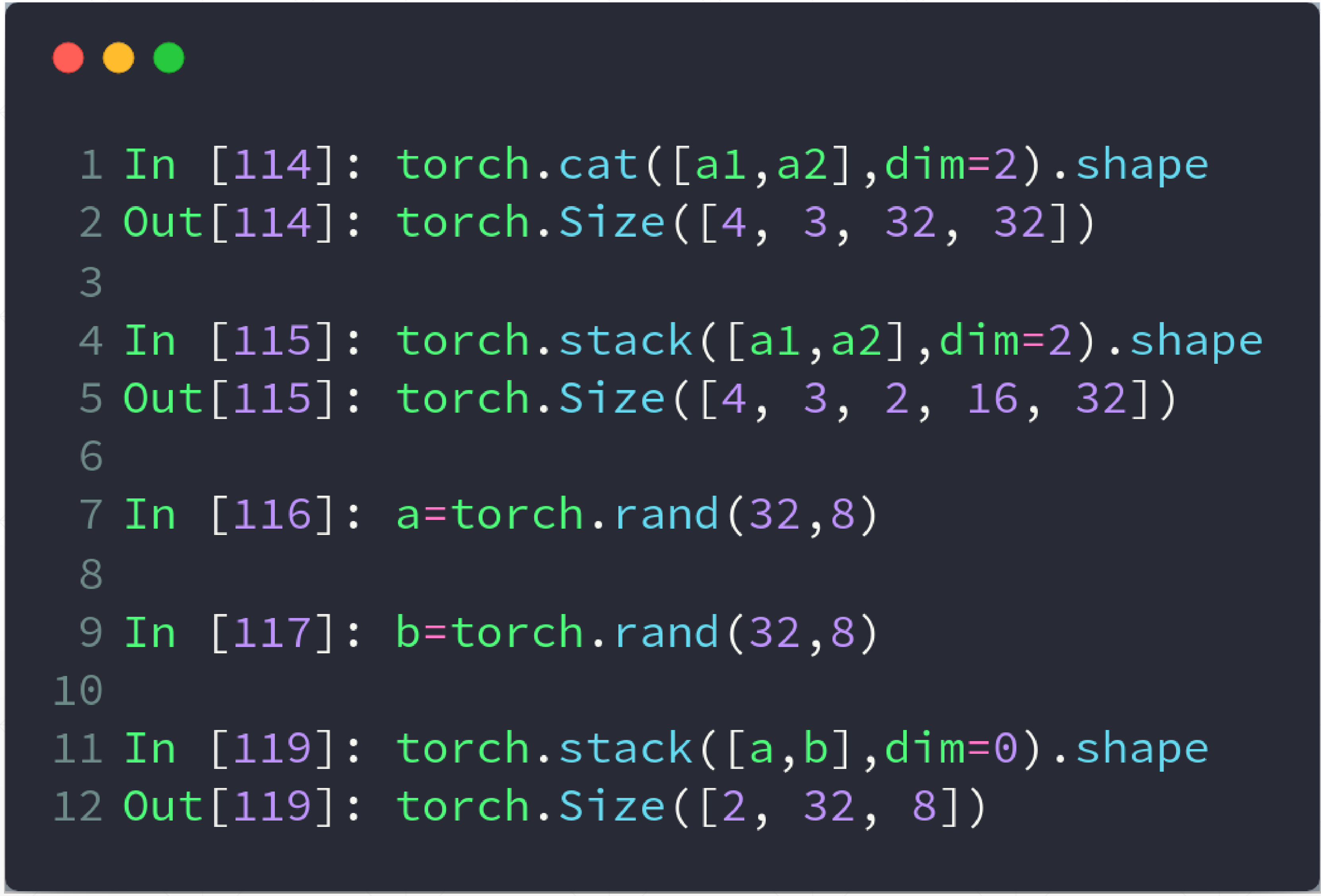
for example



```
In [103]: a1=torch.rand(4,3,32,32)
In [104]: a2=torch.rand(5,3,32,32)

In [105]: torch.cat([a1,a2],dim=0).shape
Out[105]: torch.Size([9, 3, 32, 32])

In [106]: a2=torch.rand(4,1,32,32)
In [107]: torch.cat([a1,a2],dim=0).shape
# RuntimeError: invalid argument 0:
Sizes of tensors must match except in dimension 0

In [108]: torch.cat([a1,a2],dim=1).shape
Out[108]: torch.Size([4, 4, 32, 32])

In [109]: a1=torch.rand(4,3,16,32)
In [110]: a2=torch.rand(4,3,16,32)

In [111]: torch.cat([a1,a2],dim=2).shape
Out[111]: torch.Size([4, 3, 32, 32])
```

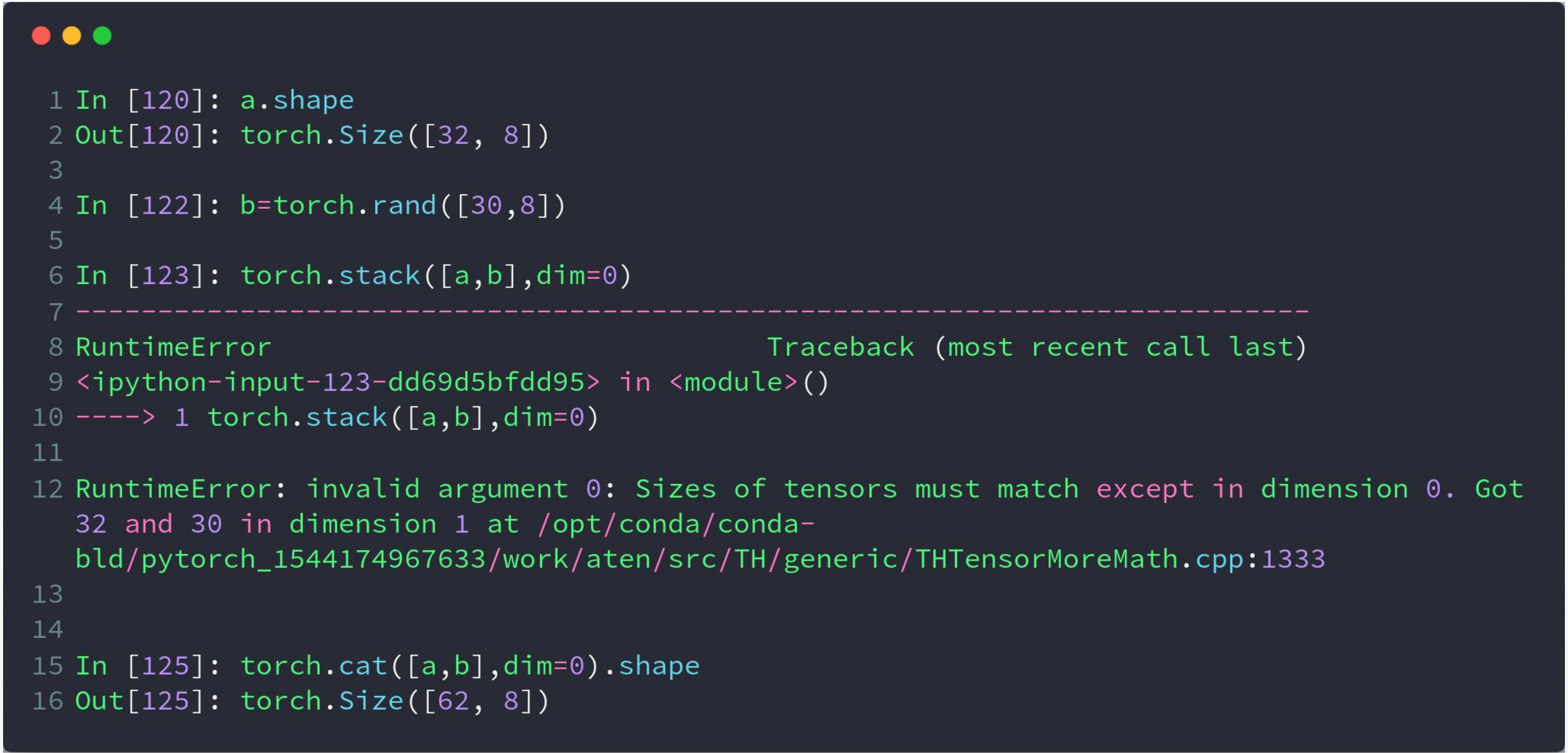
stack

create new dim



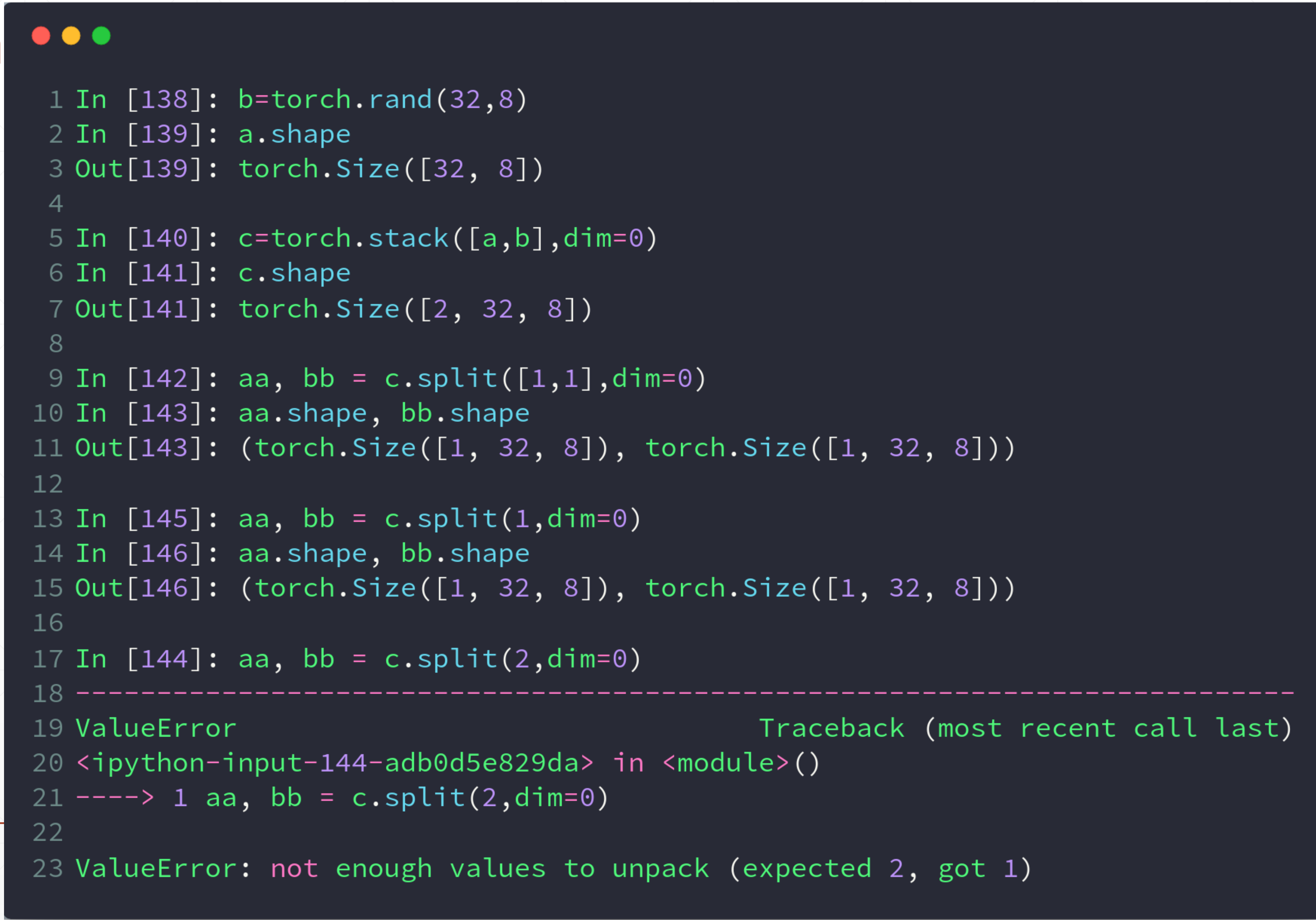
```
In [114]: torch.cat([a1,a2],dim=2).shape
Out[114]: torch.Size([4, 3, 32, 32])

In [115]: torch.stack([a1,a2],dim=2).shape
Out[115]: torch.Size([4, 3, 2, 16, 32])

In [116]: a=torch.rand(32,8)

In [117]: b=torch.rand(32,8)

In [119]: torch.stack([a,b],dim=0).shape
Out[119]: torch.Size([2, 32, 8])
```

Cat v.s. stack



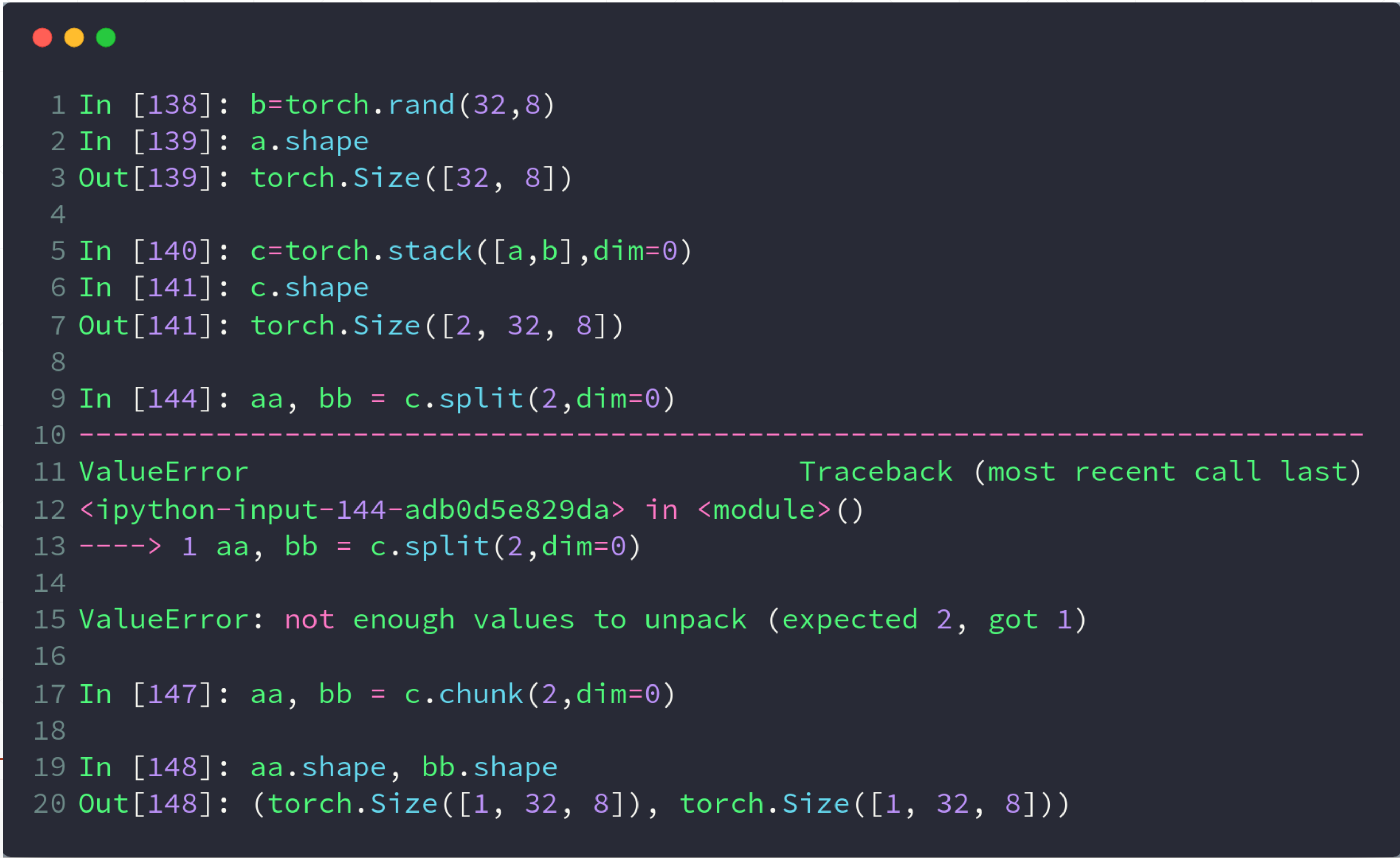
```
In [120]: a.shape
Out[120]: torch.Size([32, 8])

In [122]: b=torch.rand([30,8])

In [123]: torch.stack([a,b],dim=0)
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-123-dd69d5bfdd95> in <module>()
----> 1 torch.stack([a,b],dim=0)

RuntimeError: invalid argument 0: Sizes of tensors must match except in dimension 0. Got
32 and 30 in dimension 1 at /opt/conda/conda-
bld/pytorch_1544174967633/work/aten/src/TH/generic/THTensorMoreMath.cpp:1333


In [125]: torch.cat([a,b],dim=0).shape
Out[125]: torch.Size([62, 8])
```

Split: by len



```
In [138]: b=torch.rand(32,8)
In [139]: a.shape
Out[139]: torch.Size([32, 8])

In [140]: c=torch.stack([a,b],dim=0)
In [141]: c.shape
Out[141]: torch.Size([2, 32, 8])

In [142]: aa, bb = c.split([1,1],dim=0)
In [143]: aa.shape, bb.shape
Out[143]: (torch.Size([1, 32, 8]), torch.Size([1, 32, 8]))

In [145]: aa, bb = c.split(1,dim=0)
In [146]: aa.shape, bb.shape
Out[146]: (torch.Size([1, 32, 8]), torch.Size([1, 32, 8]))

In [144]: aa, bb = c.split(2,dim=0)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-144-adb0d5e829da> in <module>()
----> 1 aa, bb = c.split(2,dim=0)

ValueError: not enough values to unpack (expected 2, got 1)
```

Chunk: by num



```
In [138]: b=torch.rand(32,8)
In [139]: a.shape
Out[139]: torch.Size([32, 8])

In [140]: c=torch.stack([a,b],dim=0)
In [141]: c.shape
Out[141]: torch.Size([2, 32, 8])

In [144]: aa, bb = c.split(2,dim=0)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-144-adb0d5e829da> in <module>()
----> 1 aa, bb = c.split(2,dim=0)

ValueError: not enough values to unpack (expected 2, got 1)

In [147]: aa, bb = c.chunk(2,dim=0)

In [148]: aa.shape, bb.shape
Out[148]: (torch.Size([1, 32, 8]), torch.Size([1, 32, 8]))
```