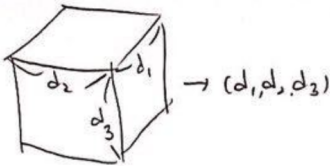
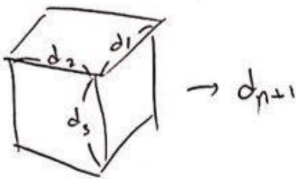


# Shape

torch.Tensor.size, torch.Tensor.stride

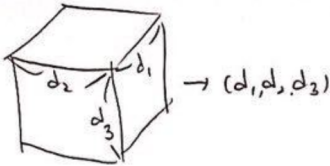
a.size() or a.stride()	
	Require
	<ul style="list-style-type: none"><li>• <math> a  = (d_1, d_2, \dots, d_k)</math></li></ul>
	Guarantees
	<ul style="list-style-type: none"><li>• <math>(d_1, d_2, \dots, d_k)</math>를 튜플로 반환</li></ul>

$$\forall \text{ft} \in \{\text{size}, \text{stride}, \}, \quad \frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash E.\text{ft}() \Rightarrow \text{shapeToTuple}(e), c}$$

a.size(n)	
	Require
	<ul style="list-style-type: none"><li>• <math> a  = (d_1, d_2, \dots, d_k)</math></li><li>• <math>0 \leq n &lt; k</math></li></ul>
	Guarantees
	<ul style="list-style-type: none"><li>• <math>d_{n+1}</math>을 숫자(int)로 반환</li><li>• <math>n</math>에 <math>-1</math>이 들어갈 수도 있음</li></ul>

$$\begin{aligned} &\sigma \vdash E \Rightarrow e, c \\ &k = \text{rank}(e) \\ &c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \\ \forall \text{ft} \in \{\text{size}, \text{stride}, \}, &\frac{}{\sigma \vdash E.\text{ft}(n) \Rightarrow e[n+1], c \cup c'} \end{aligned}$$

torch.Tensor.shape

a.shape	
	Require
	<ul style="list-style-type: none"><li>• <math> a  = (d_1, d_2, \dots, d_k)</math></li></ul>
	Guarantees
	<ul style="list-style-type: none"><li>• <math>(d_1, d_2, \dots, d_k)</math>를 튜플로 반환</li></ul>

$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash E.\text{shape} \Rightarrow \text{shapeToTuple}(e), c}$$

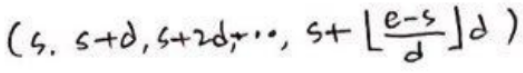
# Tensor Declarations

torch.tensor

torch.tensor(x)	
	Comment
	numpy나 파이썬 리스트로 선언된 객체를 torch에서 호환가능한 형태로 바꾸는 함수

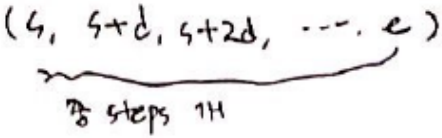
$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash \text{tensor}(E) \Rightarrow e, c}$$

torch.range, torch.arange

torch.range(start=0, end, step=1, out=None, ...), torch.arange(...)	
	Require
	<ul style="list-style-type: none"> <li>• <math>step \neq 0</math></li> <li>• <math>(end - start)/step &gt; 0</math></li> </ul>
	Guarantees
	<ul style="list-style-type: none"> <li>• <math> y  = (1 + \lfloor (end - start)/step \rfloor)</math></li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• <math>(start, start + step, start + 2 \cdot step, \dots)</math>를 반환</li> <li>• <i>out</i>-텐서 인자가 있는 함수</li> <li>• torch.arange도 똑같은 방식으로 작동함</li> </ul>

$$\forall ft \in \{\text{range}, \text{arange}\}, \quad \frac{c = \{(d \neq 0) \wedge ((e - s)/d > 0)\}}{\sigma \vdash ft(s, e, d, out = None, \dots) \Rightarrow (1 + \lfloor (e - s)/d \rfloor), c} \quad \text{Default: } s = 0, d = 1$$

torch.linspace

torch.linspace(start, end, steps=100, out=None, ...)	
	Require
	<ul style="list-style-type: none"> <li>• <math>steps \geq 0</math></li> </ul>
	Guarantees
	<ul style="list-style-type: none"> <li>• <math> y  = (steps)</math></li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• <math>(start, start + d, start + 2d, \dots, end)</math>인데 원소가 <i>steps</i>개인 텐서 반환</li> <li>• <i>out</i>-텐서 인자가 있는 함수</li> </ul>

$$\frac{c = \{(steps \geq 0)\}}{\sigma \vdash \text{linspace}(s, e, steps = 100, out = None, \dots) \Rightarrow (steps), c}$$

torch.zeros, torch.empty, torch.rand, torch.randn

torch.zeros(t1, t2, ..., tl, out=None, ...) or .empty, .rand, .randn	
	Require
	Guarantees
	<ul style="list-style-type: none"> <li>• <math> y  = (t_1, t_2, \dots, t_l)</math></li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• 입력받은 형태대로 0, uninitialized, uniformly random, gaussian random 텐서를 반환</li> <li>• <math>(t_1, t_2, \dots, t_l)</math> 입력이 하나의 튜플로 들어오는 경우도 있음</li> <li>• <i>out</i>-텐서 인자가 있는 함수</li> </ul>

$$\forall \mathbf{ft} \in \{\text{zeros, empty, rand, randn}\}, \quad \overline{\sigma \vdash \mathbf{ft}(t_1, t_2, \dots, t_l, \text{out} = \text{None}) \Rightarrow (t_1, t_2, \dots, t_l), \emptyset}$$

$$\forall \mathbf{ft} \in \{\text{zeros, empty, rand, randn}\}, \quad \overline{\sigma \vdash \mathbf{ft}((t_1, t_2, \dots, t_l), \text{out} = \text{None}) \Rightarrow (t_1, t_2, \dots, t_l), \emptyset}$$

torch.randint

torch.randint(low=0, high, shape, ..., out=None, ...)	
	Require
	<ul style="list-style-type: none"> <li>• <math>low &lt; high</math></li> <li>• <i>shape</i>가 well-defined인 텐서 shape. (스칼라 타입은 X)</li> </ul>
	Guarantees
	<ul style="list-style-type: none"> <li>• <math> y  = shape</math></li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• <i>low</i>부터 <i>high</i> - 1까지 랜덤한 정수 할당</li> <li>• <i>out</i>-텐서가 있는 함수</li> </ul>

$$\overline{\sigma \vdash \text{randint}(low = 0, high, (t_1, t_2, \dots, t_l), \dots, \text{out} = \text{None}, \dots) \Rightarrow (t_1, t_2, \dots, t_l), \{(low < high)\}}$$

torch.randperm

torch.randperm(n, out=None, ...)	
	Require
	<ul style="list-style-type: none"> <li>• <math>n \geq 0</math></li> </ul>
	Guarantees
	<ul style="list-style-type: none"> <li>• <math> y  = (n)</math></li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• <i>out</i>-텐서가 있는 함수</li> </ul>

$$\overline{\sigma \vdash \text{randperm}(n, \text{out} = \text{None}, \dots) \Rightarrow (n), \{(n \geq 0)\}}$$

## torch.full

torch.full((t1, t2, ..., tl), fill_value, out=None, ...)	
	Require
	Guarantees
	<ul style="list-style-type: none"> <li>• <math> y  = (t_1, t_2, \dots, t_l)</math></li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• 모든 원소가 <i>fill_value</i>로 채워진 텐서 반환</li> <li>• <code>empty</code>랑 비슷해보이는데, size인자를 항상 튜플로만 받음</li> <li>• <i>out</i>-텐서가 있는 함수</li> </ul>

$$\overline{\sigma \vdash \text{full}((t_1, t_2, \dots, t_l), \text{fill\_value}, \text{out} = \text{None})} \Rightarrow (t_1, t_2, \dots, t_l), \emptyset$$

## torch.normal

torch.normal(mean, std, (t1, t2, ..., tl), out=None)	
	Require
	<ul style="list-style-type: none"> <li>• <math>\text{std} &gt; 0</math></li> </ul>
	Guarantees
	<ul style="list-style-type: none"> <li>• <math> y  = (t_1, t_2, \dots, t_l)</math></li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• <code>empty</code>랑 비슷해보이는데, size인자를 항상 튜플로만 받음</li> <li>• <i>out</i>-텐서가 있는 함수</li> </ul>

$$\overline{\sigma \vdash \text{normal}(\text{mean}, \text{std}, (t_1, t_2, \dots, t_l), \text{out} = \text{None})} \Rightarrow (t_1, t_2, \dots, t_l), \{(\text{std} > 0)\}$$

## torch.Tensor.new\_empty

x.new_empty((t1, t2, ..., tl), ...)	
	Require
	Guarantees
	<ul style="list-style-type: none"> <li>• <math> y  = (t_1, t_2, \dots, t_l)</math></li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• <code>empty</code>랑 비슷해보이는데, <code>Tensor</code> 클래스에서만 사용 가능하고, <i>out</i> 인자도 없으며, size인자를 항상 튜플로만 받음</li> <li>• 텐서 <i>x</i>의 shape도 똑같이 변함</li> </ul>

$$\overline{\sigma \vdash x.\text{new\_empty}((t_1, t_2, \dots, t_l), \dots)} \Rightarrow (t_1, t_2, \dots, t_l), \emptyset$$

*x*의 shape도 똑같이  $(t_1, t_2, \dots, t_l)$ 로 변함

torch.Tensor.new\_full

x.new_full((t1, t2, ..., tl), fill_value, ...)	
	Require
	Guarantees
	<ul style="list-style-type: none"><li>• <math> y  = (t_1, t_2, \dots, t_l)</math></li></ul>
	Comment
	<ul style="list-style-type: none"><li>• new_empty랑 비슷해보이는데, fill_value 인자가 추가적으로 있음</li><li>• 텐서 x의 shape도 똑같이 변함</li></ul>

$$\frac{}{\sigma \vdash x.\text{new\_full}((t_1, t_2, \dots, t_l), \text{fill\_value} \dots) \Rightarrow (t_1, t_2, \dots, t_l), \emptyset}$$

$x$ 의 shape도 똑같이  $(t_1, t_2, \dots, t_l)$ 로 변함

torch.Tensor.clone

x.clone(...)	
	Require
	Guarantees
	<ul style="list-style-type: none"><li>• <math> y  =  x </math></li></ul>

$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash E.\text{clone}(\dots) \Rightarrow e, c}$$

torch.zeros\_like, torch.empty\_like, torch.rand\_like, torch.randn\_like

torch.zeros_like(input, ...) or .empty_like, .rand_like, .randn_like	
	Require
	Guarantees
	<ul style="list-style-type: none"><li>• <math> y  =  \text{input} </math></li></ul>
	Comment
	<ul style="list-style-type: none"><li>• 입력받은 텐서와 shape이 같은 0, uninitialized, uniformly random, gaussian random 텐서를 반환</li><li>• out-텐서 인자가 없음!</li></ul>

$$\forall \text{ft} \in \{\text{zeros\_like}, \text{empty\_like}, \text{rand\_like}, \text{randn\_like}\}, \quad \frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash \text{ft}(E, \dots) \Rightarrow e, c}$$

torch.full\_like

torch.full_like(input, fill_value, out=None, ...)	
	Require
	Guarantees
	<ul style="list-style-type: none"><li>• <math> y  =  input </math></li></ul>
	Comment
	<ul style="list-style-type: none"><li>• 입력받은 텐서와 shape이 같은 <i>fill_value</i>로 가득찬 텐서 반환</li><li>• 희한하게 이건 <i>out</i>-텐서 인자가 있음..</li></ul>

$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash \text{full\_like}(E, \text{fill\_value}, \text{out} = \text{None} \dots) \Rightarrow e, c}$$

torch.scalar\_tensor

torch.scalar_tensor(scalar, ...)	
$5 \rightarrow \text{torch.tensor}(5)$	Require
	Guarantees
	<ul style="list-style-type: none"><li>• <math> y  = ()</math></li></ul>
	Comment
	<ul style="list-style-type: none"><li>• <i>scalar</i> 값 하나만 가지는 <b>rank-0</b> 텐서 반환</li></ul>

$$\sigma \vdash \text{scalar\_tensor}(\text{scalar}, \dots) \Rightarrow (), \emptyset$$

torch.eye

torch.eye(n, m=None, out=None, ...)	
$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	Require
	<ul style="list-style-type: none"><li>• <math>n \geq 0</math></li><li>• <math>m = \text{None}</math> or <math>m \geq 0</math></li></ul>
	Guarantees
	<ul style="list-style-type: none"><li>• <math> y  = \begin{cases} (n, n) &amp; \text{if } m \text{ is } \text{None} \\ (n, m) &amp; \text{otherwise} \end{cases}</math></li></ul>
	Comment
	<ul style="list-style-type: none"><li>• 곱셈의 항등원 <math>I_n</math>를 리턴</li><li>• <i>out</i> 인자가 있는 함수</li></ul>

$$\frac{e = \text{if } m = \text{None} \text{ then } (n, n) \text{ else } (n, m) \quad c = \{(n \geq 0) \wedge (m = \text{None} \vee m \geq 0)\}}{\sigma \vdash \text{eye}(n, m, \text{out} = \text{None}, \dots) \Rightarrow e, c}$$

Same Shape, Elementwise Operators

All these builtin functions `torch.*` are used with `torch.ft(input, out=None)` that output the same shapes of the inputs.

- `round, floor, ceil`
- `exp, log, log10, log2, log1p, sigmoid`
- `sqrt, rsqrt`
- `cos, sin, tan, angle`
- `sign, neg, frac`
- `torch.Tensor.contiguous`

- 텐서 객체에서 사용 가능한 함수.
- 똑같은 내용물이지만, 메모리 상에서 원소들이 연속하도록 배치해주므로 같은 shape을 반환
- 즉, 애는 `a.contiguous()` 이런 식으로 많이 쓰임

*input* 인자는 무조건 텐서 type이어야 합니다. (스칼라 X)

$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash \text{ft}(E, \text{out} = \text{None}) \Rightarrow e, c}$$

`torch.clamp(input, min, max, out=None)` 함수는 *input* 텐서의 모든 원소가  $\min \leq \cdot \leq \max$ 가 성립하도록 만들어주는 것으로, 역시 텐서 shape이 보존됩니다.

$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash \text{clamp}(E, \min, \max, \text{out} = \text{None}) \Rightarrow e, c}$$

## Broadcasted Shape, Binary Operators

All these builtin functions `torch.*` are used with `torch.ft(input, other, out=None)` that output the same shapes of the *inputs*.

- `eq, le, lt, ge, gt`
- `mul, div, fmod, atan2`

한 가지 독특한 성질은 *input, other* 인자가 스칼라로 들어오면 []-shape 텐서로 변환된 후 계산됩니다.

- 즉, `torch.mul(1, 2).shape`은 [] 입니다.

$$\frac{\begin{array}{c} \sigma \vdash E_1 \Rightarrow e_1, c_1 \\ \sigma \vdash E_2 \Rightarrow e_2, c_2 \end{array}}{\sigma \vdash \text{ft}(E_1, E_2, \text{out} = \text{None}) \Rightarrow \text{broadcast}(e_1, e_2), c_1 \cup c_2 \cup \text{broadcastable}(e_1, e_2)}$$

The following builtin functions `torch.*` are slightly different. `torch.ft(input, other, out=None, alpha=1)` that output the same shapes of the *inputs*. (Additional *alpha* option!)

- `add, sub`
  - Calculates broadcasted  $\text{input} \pm \alpha \cdot \text{output}$

마찬가지로 *input, other* 인자로 스칼라가 들어오면 []-shape 텐서로 변환된 후 계산됩니다.

$$\frac{\begin{array}{c} \sigma \vdash E_1 \Rightarrow e_1, c_1 \\ \sigma \vdash E_2 \Rightarrow e_2, c_2 \end{array}}{\sigma \vdash \text{ft}(E_1, E_2, \text{out} = \text{None}, \alpha = 1) \Rightarrow \text{broadcast}(e_1, e_2), c_1 \cup c_2 \cup \text{broadcastable}(e_1, e_2)}$$

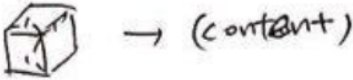
torch.pow

torch.pow(input, exponent, out=None)	
	Require
	<ul style="list-style-type: none"><li>• [<i>input</i> or <i>exponent</i> are scalar] or <i>broadcastable</i>(<i> input </i>, <i> exponent </i>)</li></ul>
	Guarantees
	<ul style="list-style-type: none"><li>• <i>broadcast</i>(<i> input </i>, <i> exponent </i>)</li></ul>
	Comment
	<ul style="list-style-type: none"><li>• elementwise binary operator라는 점에서 mul과 비슷하나, 인자 이름(<i>exponent</i>)이 달라서 따로 뺐습니다. (<code>pow(a, exponent=b)</code>같은 문제..)</li><li>• 마찬가지로 인자에 스칼라가 들어오면 []-shape 텐서로 변환된 후 계산됩니다.</li><li>• <i>out</i> 인자가 있는 함수</li></ul>

$$\begin{array}{c} \sigma \vdash E_1 \Rightarrow e_1, c_1 \\ \sigma \vdash E_2 \Rightarrow e_2, c_2 \\ \hline \sigma \vdash \text{pow}(E_1, E_2, \text{out} = \text{None}) \Rightarrow \text{broadcast}(e_1, e_2), c_1 \cup c_2 \cup \text{broadcastable}(e_1, e_2) \end{array}$$

Indexing

torch.Tensor.item

a.item()	
	Require
	<ul style="list-style-type: none"><li>• <math> a  = ()</math> or <math> a  = (1, 1, 1, \dots, 1)</math></li></ul>
	Guarantees
	<ul style="list-style-type: none"><li>• <math> y  = e_n</math></li></ul>
	Comment
	<ul style="list-style-type: none"><li>• Singular element tensor의 원소(스칼라 타입으로 반환)</li></ul>

$$\begin{array}{c} \sigma \vdash E \Rightarrow e, c \\ k = \text{rank}(e) \\ c' = \{(\forall i = 1, 2, \dots, k, e[i] = 1)\} \\ \hline \sigma \vdash E.\text{item}() \Rightarrow e_n, c \cup c' \end{array}$$



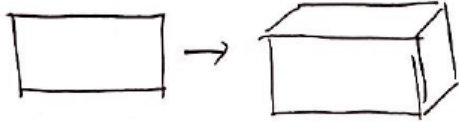
# Shape Polymorphism

torch.Tensor.view

a.view(n1, n2, ..., nl)	
	Require
	<ul style="list-style-type: none"> <li>• <math> a  = (d_1, d_2, \dots, d_k)</math></li> <li>• <math>reshapable((d_1, d_2, \dots, d_k), (n_1, n_2, \dots, n_l))</math></li> </ul>
	Guarantees
	<ul style="list-style-type: none"> <li>• <math>(n_1, n_2, \dots, n_l)</math> as tensor</li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• 단, <math>n_1, n_2, \dots, n_l</math>이 하나의 튜플로 입력이 들어올 수도 있음</li> </ul>

$$\frac{\sigma \vdash \text{reshape}(E, n_1, n_2, \dots, n_l) \Rightarrow e, c}{\sigma \vdash E.\text{view}(n_1, n_2, \dots, n_l) \Rightarrow e, c}$$

torch.Tensor.expand

a.expand(n1, n2, ..., nl)	
	Require
	<ul style="list-style-type: none"> <li>• <math> a  = (d_1, d_2, \dots, d_k)</math></li> <li>• <math>k \leq l</math></li> <li>• <math>\forall i = 1, 2, \dots, l - k, (n_i &gt; 0)</math></li> <li>• <math>\forall i = l - k + 1, l - k + 2, \dots, l, [(n_i = -1) \text{ or } ((n_i &gt; 0) \text{ and } ((d_{i-(l-k)} = 1) \text{ or } (d_{i-(l-k)} = n_i)))]</math></li> </ul>
	Guarantees
	<ul style="list-style-type: none"> <li>• <math>(m_1, m_2, \dots, m_l)</math> as tensor where <ul style="list-style-type: none"> <li>- <math>m_1 = n_1, m_2 = n_2, \dots, m_{l-k} = n_{l-k}</math></li> <li>- <math>m_{l-k+i} = \text{if } (n_{l-k+i} = -1) \text{ then } d_i \text{ else } n_{l-k+i} \text{ for the rests}</math></li> </ul> </li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• 일방향 broadcast 함수; <math>(n_1, n_2, \dots, n_l)</math> shape이 목표</li> <li>• <math>n_i = -1</math>인 경우, 본래 크기만큼 그대로 유지</li> <li>• 단, <math>n_1, n_2, \dots, n_l</math>이 하나의 튜플로 입력이 들어올 수도 있음</li> </ul>

$$\sigma \vdash E \Rightarrow e, c$$

$$k = \text{rank}(e)$$

$$m_1 = n_1, m_2 = n_2, \dots, m_{l-k} = n_{l-k}$$

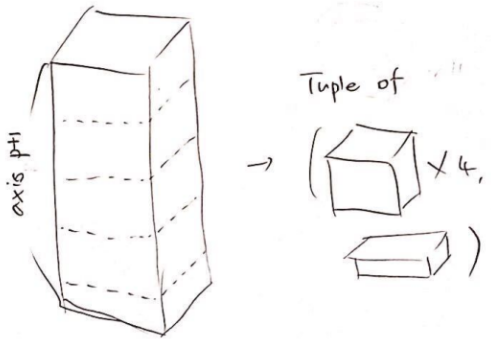
$$\forall i \in \{1, 2, \dots, k\}, m_{l-k+i} = \text{if } (n_{l-k+i} = -1) \text{ then } d_i \text{ else } n_{l-k+i}$$

$$c_{base} = \{(k \leq l) \wedge (n_1 > 0) \wedge (n_2 > 0) \wedge \dots \wedge (n_{l-k} > 0)\}$$

$$c_{expandable} = \{(\forall i \in \{1, 2, \dots, k\}, (n_{l-k+i} = -1) \vee [(n_{l-k+i} > 0) \wedge ((d_i = 1) \vee (d_i = n_{l-k+i}))])\}$$

$$\sigma \vdash E.\text{expand}(n_1, n_2, \dots, n_l) \Rightarrow (m_1, m_2, \dots, m_l), c \cup c_{base} \cup c_{expandable}$$

## torch.split

torch.split(tensor, split_size_or_section, dim=0)	
	Require
	<ul style="list-style-type: none"> <li>• <math> tensor  = (d_1, d_2, \dots, d_k)</math></li> <li>• <math>k \geq 1</math></li> <li>• <math>0 \leq dim &lt; k</math></li> </ul>
	Guarantees
	<ul style="list-style-type: none"> <li>• 아래 proof tree와 같이 <math>dim + 1</math>번째 axis가 최대 <math>split\_size\_or\_section</math>개의 원소를 가지도록 <math>\lceil d_{dim+1}/\cdot \rceil</math>개 튜플 형태로 반환</li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• Concat의 반대 역할 함수</li> <li>• Divisible 여부 assert하지 않음</li> <li>• 학습 및 테스트 데이터를 배치단위로 쪼개는 용도로 쓰일 것으로 추측</li> </ul>

$$\sigma \vdash E \Rightarrow e, c$$

$$k = \text{rank}(e)$$

$$e_1 = e[1:p]@ (n) @ e[p+2:k]$$

$$e_2 = e[1:p]@ (n) @ e[p+2:k]$$

...

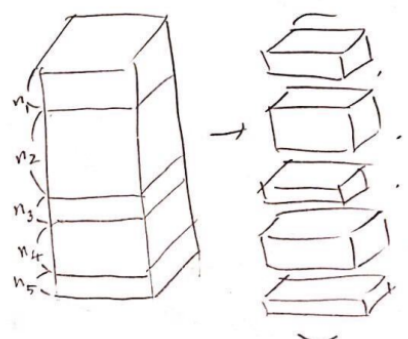
$$e_{l-1} = e[1:p]@ (n) @ e[p+2:k]$$

$$e_l = e[1:p]@ (n') @ e[p+2:k] \quad \text{where } e[p+1] = n(l-1) + n', 0 < n' \leq n$$

$$c' = \{(k \geq 1) \wedge (0 \leq p < k)\}$$

$$\sigma \vdash \text{split}(E, n, p=0) \Rightarrow (e_1, e_2, \dots, e_l), c \cup c'$$

$l$ -원소 tuple 형태로 반환

torch.split(input, [n1, n2, ..., nl], dim=0)	
	Require
	<ul style="list-style-type: none"> <li>• <math> input  = (d_1, d_2, \dots, d_k)</math></li> <li>• <math>k \geq 1</math></li> <li>• <math>0 \leq dim &lt; k</math></li> <li>• <math>d_{dim+1} = n_1 + n_2 + \dots + n_l</math></li> </ul>
	Guarantees
	<ul style="list-style-type: none"> <li>• 아래 proof tree와 같이 <math>dim + 1</math>번째 axis의 크기가 <math>n_1, n_2, \dots, n_l</math>인 <math>l</math>개의 텐서 튜플을 반환</li> <li>• <math>n_i</math>의 합과 <math>d_{dim+1}</math>이 같은지 assert</li> </ul>

$$\sigma \vdash E \Rightarrow e, c$$

$$k = \text{rank}(e)$$

$$e_1 = e[1:p]@ (n_1) @ e[p+2:k]$$

$$e_2 = e[1:p]@ (n_2) @ e[p+2:k]$$

...

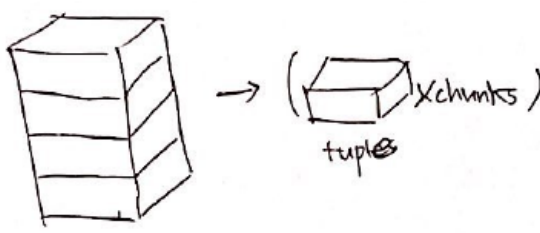
$$e_l = e[1:p]@ (n_l) @ e[p+2:k]$$

$$c' = \{(k \geq 1) \wedge (0 \leq x < k) \wedge (e[p+1] = n_1 + n_2 + \dots + n_l)\}$$

$$\sigma \vdash \text{split}(E, [n_1, n_2, \dots, n_l], p=0) \Rightarrow (e_1, e_2, \dots, e_l), c \cup c'$$

$l$ -원소 tuple 형태로 반환

## torch.chunk

torch.chunk(input, chunks, dim=0)	
	Require
	<ul style="list-style-type: none"> <li>• <math> input  = (d_1, d_2, \dots, d_k)</math></li> <li>• <math>k \geq 1</math></li> <li>• <math>chunks &gt; 0</math></li> <li>• <math>0 \leq dim &lt; k</math></li> </ul>
	Guarantees
	<ul style="list-style-type: none"> <li>• Proof tree와 같이 최소 <math>chunks</math>개로 <math>dim</math> 축이 잘라진 텐서 튜플 반환</li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• <code>split</code>과 비슷하나 쪼개는 개수를 명시한다는 점이 다름</li> </ul>

$$\sigma \vdash E \Rightarrow e, c$$

$$k = \mathbf{rank}(e)$$

$$d = \min\{e[p+1], chunks\}$$

$$n = \lceil e[p+1]/d \rceil$$

$$e_1 = e[1:p] @ (n) @ e[p+2:k]$$

$$e_2 = e[1:p] @ (n) @ e[p+2:k]$$

...

$$e_{l-1} = e[1:p] @ (n) @ e[p+2:k]$$

$$e_l = e[1:p] @ (n') @ e[p+2:k] \quad \text{where } e[p+1] = n(l-1) + n', 0 < n' \leq n$$

$$c' = \{(k \geq 1) \wedge (chunks > 0) \wedge (0 \leq p < k)\}$$

$$\sigma \vdash \mathbf{chunk}(E, chunks, p = 0) \Rightarrow (e_1, e_2, \dots, e_l), c \cup c'$$

$l$ -원소 tuple 형태로 반환

## Dynamic Operations (Cannot Predict Statically)

### torch.nonzero

torch.nonzero(input, out=None, as_tuple=False)	
	Require
	Guarantees
	<ul style="list-style-type: none"> <li>• if <i>as_tuple</i>, then <math> y  = (\mathbf{countNonzeros}(\mathbf{input}), \mathbf{rank}( \mathbf{input} ))</math></li> <li>• otherwise, then <math> y </math> is <math>\mathbf{rank}( \mathbf{input} )</math>-tuple of tensors shaped <math>(\mathbf{countNonzeros}(\mathbf{input}))</math></li> </ul>
	Comment
	<ul style="list-style-type: none"> <li>• <code>countNonzeros</code>는 주어진 텐서에서 0이 아닌 항의 개수를 구하는 것</li> <li>• Nonzero 인덱스 번호들을 반환함.</li> <li>• <i>out</i>-텐서 인자가 있는 함수</li> </ul>

$$\sigma \vdash E_1 \Rightarrow e_1, c_1$$

$$\sigma \vdash E_2 \Rightarrow e_2, c_2$$

$$\sigma \vdash \mathbf{pow}(E_1, E_2, out = None) \Rightarrow \mathbf{broadcast}(e_1, e_2), c_1 \cup c_2 \cup \mathbf{broadcastable}(e_1, e_2)$$

Example Codes:

```
print(torch.nonzero(torch.tensor([[0, 1, 0], [0, 1, 1]])))
```

```
# output: tensor([[0, 1], [1, 1], [1, 2]])
```

```
print(torch.nonzero(torch.tensor([[0, 1, 0], [0, 1, 1]]), as_tuple=True))  
# output: (tensor([0, 1, 1]), tensor([1, 1, 2]))
```