PyTorch NumPy-like Functions Cheat Sheet

PyTorch (`https://pytorch.org/`) is well-known deep learning framework. There are good reasons to do basic NumPy operations using PyTorch. Therefore, this cheat sheet covers functions that are similar.

Loading PyTorch and tensor basics

```
# loading PyTorch
import torch

# defining a tensor
torch.tensor((values))

# define data type
torch.tensor((values), dtype=torch.int16)

# converting a NumPy array to a PyTorch tensor
torch.from_numpy(numpyArray)

# create a tensor of zeros
torch.zeros((shape))
torch.zeros_like(other_tensor)

# create a tensor of ones
torch.ones((shape))
torch.ones_like(other_tensor)

# create an idenity matrix
torch.eye(numberOfRows)

# create tensor with same values
torch.full((shape), value)
torch.full_like(other_tensor,value)

# create an empty tensor
torch.empty((shape))
torch.empty_like(other_tensor)

# create sequences
torch.arange(startNumber, endNumber, stepSize)
torch.linspace(startNumber, endNumber, stepSize)
torch.logspace(startNUmber, endNumber, stepSize)

# concatenate tensors
torch.cat((tensors), axis)

# split tensors into sub-tensors
torch.split(tensor, splitSize)

# (un)squeeze tensor
torch.squeeze(tensor, dimension)
torch.unsqueeze(tensor, dim)

# reshape tensor
torch.reshape(tensor, shape)

# transpose tensor
torch.t(tensor) # 1D and 2D tensors
torch.transpose(tensor, dim0, dim1)
```

Random numbers

```
# set seed
torch.manual_seed(seed)

# generate a tensor with random numbers
# of interval [0,1)
torch.rand(size)
torch.rand_like(other_tensor)

# generate a tensor with random integer numbers
# of interval [lowerInt,higherInt]
torch.randint(lowerInt, higherInt, (tensor_shape))
torch.randint_like(other_tensor, lowerInt, higherInt)

# generate a tensor of random numbers drawn
# from a normal distribution (mean=0, var=1)
torch.randn((size))
torch.randn_like(other_tensor)

# random permuation of integers
# range [0,n-1)
torch.randperm()
```

Math (element-wise)

```
# basic operations
torch.abs(tensor)
torch.add(tensor, tensor2) # or tensor+scalar
torch.div(tensor, tensor2) # or tensor/scalar
torch.mult(tensor,tensor2) # or tensor*scalar
torch.sub(tensor, tensor2) # or tensor-scalar
torch.ceil(tensor)
torch.floor(tensor)
torch.remainder(tensor, devisor) #or torch.fmod()
torch.sqrt(tensor)

# trigonometric functions
torch.acos(tensor)
torch.asin(tensor)
torch.atan(tensor)
torch.atan2(tensor)
torch.cos(tensor)
torch.cosh(tensor)
torch.sin(tensor)
torch.sinh(tensor)
torch.tan(tensor)
torch.tanh(tensor)

# exponentials and logarithms
torch.exp(tensor)
torch.expm1(tensor) # exp(input-1)
torch.log(tensor)
torch.log10(tensor)
torch.log1p(tensor) # log(1+input)
```

```
torch.log2(tensor)

# other
torch.erfc(tensor) # error function
torch.erfinv(tensor) # inverse error function
torch.round(tensor) # round to full integer
torch.power(tensor, power)
```

Math (not element-wise)

```
torch.argmax(tensor)
torch.argmin(tensor)
torch.max(tensor)
torch.min(tensor)
torch.mean(tensor)
torch.median(tensor)
torch.norm(tensor, norm)
torch.prod(tensor) # product of all elements
torch.std(tensor)
torch.sum(tensor)
torch.unique(tensor)
torch.var(tensor)
torch.cross(tensor1,tensor2)
torch.cartesian_prod(tensor1, tensor2, ...)
torch.einsum(equation,tensor)
```



```
torch.tensordot(tensor1,tensor2)
torch.cholesky(tensor)
torch.cholesky_torch(tensor)
torch.dot(tensor1, tensor2)
torch.eig(tensor)
torch.inverse(tensor)
torch.det(tensor)
torch.pinverse(tensor) # pseudo-inverse
```

Other

```
torch.isinf(tensor)
torch.sort(tensor)
torch.fft(tensor, signal_dim)
torch.ifft(tensor, signal_dim)
torch.rfft(tensor, signal_dim)
torch.rifft(tensor, signal_dim)
torch.stft(tensor, n_fft)
torch.bincount(tensor)
torch.diagonal(tensor)
torch.flatten(tensor, start_dim)
torch.rot90(tensor)
torch.histc(tensor)
torch.trace(tensor)
torch.svd(tensor)
```