1. Obtain a tensor containing only zeroes from the given tensor

```
pattern = torch.tensor([
    [1, 1, 1, 1],
    [1, 0, 0, 1],
    [1, 0, 0, 1],
    [1, 1, 1, 1]
])
zeros_only = pattern[pattern == 0]
```

2. Create a Numpy array of shape (1,3,3) using PyTorch

```
numpy_array = torch.randn(1, 3, 3).numpy()
```

3. Create two random (2,2,2) tensors and find the max, min, mean, std of their product (matrix multiplication)

```
a = torch.randn(2, 2, 2)
b = torch.randn(2, 2, 2)
product = a @ b  # matrix multiplication for last two dimensions
print("Max:", product.max().item())
print("Min:", product.min().item())
print("Mean:", product.mean().item())
print("Std:", product.std().item())
```

4. Convert a 16x16 tensor into 1x256 tensor

```
tensor_16x16 = torch.randn(16, 16)
tensor_1x256 = tensor_16x16.view(1, 256)
print("Reshaped tensor shape:", tensor_1x256.shape)
```

4. Convert a 16x16 tensor into 1x256 tensor

```
tensor_16x16 = torch.randn(16, 16)
tensor_1x256 = tensor_16x16.view(1, 256)
print("Reshaped tensor shape:", tensor_1x256.shape)
```

5. Given two tensors x and Y, find the coefficients that best model the linear relationship `Y = ax + b` (Linear Regression)

```
x = torch.randn(10, 1)
y = 3 * x + 2 + 0.1 * torch.randn(10, 1)  # example relationship

X_ones = torch.cat([x, torch.ones_like(x)], dim=1)  # add bias term
coeffs, _ = torch.lstsq(y, X_ones)  # returns [a, b]
a, b = coeffs[:2]
print(f"Estimated coefficients: a = {a.item()}, b = {b.item()}")
```