Nama : Hurin Salimah
NIM : 1103200021

Input

```
# impor pustaka PyTorch
import torch

# membuat tensor
X = torch.rand(size=(7, 7))
# menampilkan nilai tensor 'X' dan bentuk dari tensor 'x'
X, X.shape
```

Output

```
(tensor([[0.8995, 0.8827, 0.5145, 0.8651, 0.5443, 0.7721, 0.9787],
         [0.9351, 0.0749, 0.8490, 0.5835, 0.4602, 0.4099, 0.8256],
         [0.5250, 0.6182, 0.3518, 0.4554, 0.3386, 0.2471, 0.1130],
         [0.1395, 0.2869, 0.5954, 0.3157, 0.2483, 0.6229, 0.5268],
         [0.9709, 0.9805, 0.1034, 0.8120, 0.1786, 0.0402, 0.7267],
         [0.1530, 0.4426, 0.7974, 0.8469, 0.3046, 0.8965, 0.8701],
         [0.1740, 0.9232, 0.6349, 0.8756, 0.1303, 0.6409, 0.1659]]),
 torch.Size([7, 7]))
```

Input

```
# membuat tensor acak baru
Y = torch.rand(size=(1, 7))
# menunjukkan operasi perkalian matriks antara tensor 'X' dan transpose
dari tensor'y'
Z = torch.matmul(X, Y.T) # no error because of transpose
# menampilkan nilai dari tensor hasil perkalian matriks
Z, Z.shape
```

Output

```
(tensor([[2.4903],
         [2.2991],
         [1.2449],
         [1.2928],
         [1.6896],
         [1.9207],
         [1.4432]]),
 torch.Size([7, 1]))
```

Input

```
# mengatur benih untuk generator nomor acak PyTorch
torch.manual_seed(0)
# membuat tensor acak
X = torch.rand(size=(7, 7))
# membuat tensor acak
Y = torch.rand(size=(1, 7))

# melakukan operasi perkalian matriks
Z = torch.matmul(X, Y.T)
# menampilkan nilai dari tensor hasil perkalian matriks
Z, Z.shape
```

Output

```
(tensor([[1.8542],
         [1.9611],
         [2.2884],
         [3.0481],
         [1.7067],
         [2.5290],
         [1.7989]]),
 torch.Size([7, 1]))
```

Input

```
# mengatur benih untuk generator nomor acakyang terkait dengan GPU
torch.cuda.manual_seed(1234)
```

Input

```
# mengatur benih untuk generator nomor acak pada CPU untuk PyTorch
torch.manual_seed(1234)

# memeriksa perangkat GPU tersedia atau tidak untuk operasi selanjutnya
device = "cuda" if torch.cuda.is_available() else "cpu"
# mencetak perangkat yang akan digunakan
print(f"Device: {device}")

# membuat tensor acak & mengirim ke perangkat yang ditentukan
sebelumnya
tensor_A = torch.rand(size=(2,3)).to(device)
# membuat tensor acak dan mengirim nya ke perangkat yang sama
tensor_B = torch.rand(size=(2,3)).to(device)
tensor_A, tensor_B
```

Output

```
Device: cpu
(tensor([[0.0290, 0.4019, 0.2598],
         [0.3666, 0.0583, 0.7006]]),
 tensor([[0.0518, 0.4681, 0.6738],
         [0.3315, 0.7837, 0.5631]]))
```

Input

```
# menghasilkan transpose sebelum perkalian
tensor_C = torch.matmul(tensor_A, tensor_B.T)
# mencetak tensor hasil dari operasi perkalian matriks
tensor_C, tensor_C.shape
```

Output

```
(tensor([[0.3647, 0.4709],
         [0.5184, 0.5617]]),
 torch.Size([2, 2]))
```

Input

```
# mencari nilai maksimum dari seluruh elemen yang ada dalam tensor
max = torch.max(tensor_C)

# mencari nilai minimum dari seluruh elemen yang ada dalam tensor
min = torch.min(tensor_C)
```

```
# menyimpan niali maks dan min dari tensor masing masing
max, min
```

Output

```
(tensor(0.5617), tensor(0.3647))
```

Input

```
# menemukan indeks dari elemen dengan nilai maks dalam tensor
arg_max = torch.argmax(tensor_C)

# menemukan indeks dari elemen dengan nilai min dalam tensor
arg_min = torch.argmin(tensor_C)
# menyimpan indeks dari elemen dengan nilai maks dan min dalam tensor
arg_max, arg_min
```

Output

```
(tensor(3), tensor(0))
```

Input

```
# menetapkan nilai seed untuk generator bilangan acak pada Torch
torch.manual_seed(7)

# membuat tensor acak
tensor_D = torch.rand(size=(1, 1, 1, 10))

# menghilangkan dimensi
tensor_E = tensor_D.squeeze()

# mencetak tensor D
print(tensor_D, tensor_D.shape)
# mencetak tensor E
print(tensor_E, tensor_E.shape)
```

Output

```
tensor([[[[0.5349, 0.1988, 0.6592, 0.6569, 0.2328, 0.4251, 0.2071,
0.6297,
           0.3653, 0.8513]]]]) torch.Size([1, 1, 1, 10])
tensor([0.5349, 0.1988, 0.6592, 0.6569, 0.2328, 0.4251, 0.2071, 0.6297,
0.3653,
        0.8513]) torch.Size([10])
```