```
#Nama: Dilara Kynta Putri Raflita
#NIM: 110324059
#Kelas: TK44G4


# Import torch
import torch

# Create random tensor
X = torch.rand(size=(7, 7))
X, X.shape
```

```
     (tensor([[0.9053, 0.9359, 0.5312, 0.0771, 0.5442, 0.7962, 0.8017],
              [0.3721, 0.8758, 0.9824, 0.1975, 0.3119, 0.6905, 0.5794],
              [0.2105, 0.9102, 0.1377, 0.0014, 0.3083, 0.1548, 0.6241],
              [0.8353, 0.6046, 0.9929, 0.3937, 0.5202, 0.5383, 0.0576],
              [0.6012, 0.5734, 0.1013, 0.0979, 0.8484, 0.0973, 0.2103],
              [0.8406, 0.3411, 0.2940, 0.2372, 0.1509, 0.3736, 0.4853],
              [0.5733, 0.4947, 0.5735, 0.7840, 0.9372, 0.8296, 0.4561]]),
      torch.Size([7, 7]))
```

pada kode tersebut mendefinisikan dalam membuat tensor dengan bentuk (ukuran) 7x7 yang diisi dengan nilai acak dari distribusi seragam (antara 0 dan 1). Fungsi torch.rand digunakan untuk menghasilkan nilai-nilai acak dari distribusi seragam di antara 0 (inklusif) dan 1 (eksklusif).

```
# Create another random tensor
Y = torch.rand(size=(1, 7))
# Z = torch.matmul(X, Y) # will error because of shape issues
Z = torch.matmul(X, Y.T) # no error because of transpose
Z, Z.shape
```

```
     (tensor([[1.9077],
              [1.6274],
              [0.8645],
              [2.1657],
              [1.2712],
              [1.4453],
              [2.1433]]),
      torch.Size([7, 1]))
```

kode ini membuat tensor acak baru Y dengan bentuk 1x7, dan kemudian mencoba melakukan perkalian matriks antara tensor X (7x7) dan Y. Awalnya, terdapat kesalahan karena masalah bentuk tensor, namun solusinya adalah dengan mentransposisi tensor Y sebelum melakukan perkalian matriks.

```
# Set manual seed
torch.manual_seed(0)

# Create two random tensors
X = torch.rand(size=(7, 7))
Y = torch.rand(size=(1, 7))

# Matrix multiply tensors
Z = torch.matmul(X, Y.T)
Z, Z.shape
```

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

kode tersebut membuat tensor acak X dengan bentuk 7x7 menggunakan fungsi torch.rand. Tensor ini diisi dengan nilai-nilai acak dari distribusi seragam antara 0 dan 1. kemudian membuat tensor acak Y dengan bentuk 1x7 menggunakan fungsi torch.rand. Tensor ini juga diisi dengan nilai-nilai acak dari distribusi seragam antara 0 dan 1. hasil output yang diberikan adalah tensor z yang merupakan hasil dari perkalian matriks antara 'X' dan 'Y.T'

```
# Set random seed on the GPU
torch.cuda.manual_seed(1234)
```

Kode torch.cuda.manual_seed(1234) digunakan untuk menetapkan manual seed pada generator bilangan acak PyTorch untuk perangkat GPU (CUDA). yang artinya ketika operasi acak dilakukan pada GPU, hasil acaknya akan dapat direproduksi, asalkan seed ini digunakan secara konsisten.

```
# Set random seed
torch.manual_seed(1234)

# Check for access to GPU
device = "cuda" if torch.cuda.is_available() else "cpu"
print(f"Device: {device}")

# Create two random tensors on GPU
tensor_A = torch.rand(size=(2,3)).to(device)
tensor_B = torch.rand(size=(2,3)).to(device)
tensor_A, tensor_B
```

```
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]]))
```

device = "cuda" if torch.cuda.is_available() else "cpu": Memeriksa apakah perangkat GPU (CUDA) tersedia. Jika CUDA tersedia, maka device diatur sebagai "cuda", jika tidak, maka diatur sebagai "cpu". dapat dilihat bahwa hasil output nya adalah 'cpu'. kemudian output juga mengeluarkan nilai tensor A dan tensor B yang mana kedua tensor ini berada di perangkat yang sama seperti yang ditentukan oleh variabel 'device'.

```
# Perform matmul on tensor_A and tensor_B
# tensor_C = torch.matmul(tensor_A, tensor_B) # won't work because of shape error
tensor_C = torch.matmul(tensor_A, tensor_B.T)
tensor_C, tensor_C.shape
```

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

output yang dihasilkan yaitu tensor_c yang merupakan hasil dari perkalian matriks antara tensor_A (2x3) dan transposisi dari tensor_B (3x2), sehingga bentuk tensor_C menjadi 2x2.

```
# Find max
max = torch.max(tensor_C)

# Find min
min = torch.min(tensor_C)
max, min
```

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

kode tersebut menggunakan fungsi PyTorch untuk menemukan nilai maksimum dan minimum dalam tensor tensor_C.

max = torch.max(tensor_C): Menggunakan fungsi torch.max() untuk menemukan nilai maksimum dalam tensor tensor_C.

min = torch.min(tensor_C): Menggunakan fungsi torch.min() untuk menemukan nilai minimum dalam tensor tensor_C.

Hasil output tsb yaitu variabel max yg berisi nilai maksimum dari seluruh elemen dalam tensor_C, dan variabel min berisi nilai minimum dari seluruh elemen dalam tensor_C.

```
# Find arg max
arg_max = torch.argmax(tensor_C)

# Find arg min
arg_min = torch.argmin(tensor_C)
arg_max, arg_min
```

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

Hasil output tersebut merupakan arg_max dan arg_min yaitu indeks dari nilai maksimum dan minimum dalam tensor_C

```
# Set seed
torch.manual_seed(7)

# Create random tensor
tensor_D = torch.rand(size=(1, 1, 1, 10))

# Remove single dimensions
tensor_E = tensor_D.squeeze()

# Print out tensors
print(tensor_D, tensor_D.shape)
```

```
print(tensor_E, tensor_E.shape)
```

```
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])
```

torch.manual_seed(7): Menetapkan manual seed pada generator bilangan acak PyTorch untuk memastikan hasil acak dapat direproduksi.

tensor_D = torch.rand(size=(1, 1, 1, 10)): Membuat tensor acak tensor_D dengan bentuk (shape) (1, 1, 1, 10).

tensor_E = tensor_D.squeeze(): Menggunakan metode squeeze() untuk menghilangkan dimensi yang memiliki ukuran 1 dari tensor_D dan tensor_D akan menghasilkan tensor baru tensor_E dengan bentuk yang lebih sederhana.

kemudian hasil outputnya adalah nilai dari tensor_D dan bentuknya serta nilai tensor tensor_E dan bentuknya setelah operasi squeeze().