In [26]:

```
# @title 0. Checking GPU Availability for TensorFlow

import tensorflow as tf

print(tf.config.list_physical_devices("GPU"))
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

In [27]:

```
# @title # 1. Create a vector, scalar, matrix and tensor with values of your choosing usi
ng tf.constant()

import tensorflow as tf

# Vector
vector = tf.constant([1, 2, 3, 4], dtype=tf.float32)

# Scalar
scalar = tf.constant(5, dtype=tf.int32)

# Matrix
matrix = tf.constant([[1, 2], [3, 4]], dtype=tf.float32)

# Tensor
tensor = tf.constant([[[1, 2], [3, 4]], [[5, 6], [7, 8]]], dtype=tf.float32)

print("Vector:", vector)
print("Scalar:", scalar)
print("Matrix:", matrix)
print("Tensor:", tensor)
```

```
Vector: tf.Tensor([1. 2. 3. 4.], shape=(4,), dtype=float32)
Scalar: tf.Tensor(5, shape=(), dtype=int32)
Matrix: tf.Tensor(
[[1. 2.]
 [3. 4.]], shape=(2, 2), dtype=float32)
Tensor: tf.Tensor(
[[[1. 2.]
  [3. 4.]]

 [[5. 6.]
  [7. 8.]]], shape=(2, 2, 2), dtype=float32)
```

In [28]:

```
# @title 2. Find the shape, rank and size of the tensors you created in 1.

print("Vector:")
print("Shape:", vector.shape)
print("Rank:", tf.rank(vector))
print("Size:", tf.size(vector))

print("\nScalar:")
print("Shape:", scalar.shape)
print("Rank:", tf.rank(scalar))
print("Size:", tf.size(scalar))

print("\nMatrix:")
print("Shape:", matrix.shape)
print("Rank:", tf.rank(matrix))
print("Size:", tf.size(matrix))

print("\nTensor:")
print("Shape:", tensor.shape)
print("Rank:", tf.rank(tensor))
print("Size:", tf.size(tensor))
```

```
Vector:
Shape:  (4,)
Rank: tf.Tensor(1, shape=(), dtype=int32)
Size: tf.Tensor(4, shape=(), dtype=int32)

Scalar:
Shape:  ()
Rank: tf.Tensor(0, shape=(), dtype=int32)
Size: tf.Tensor(1, shape=(), dtype=int32)

Matrix:
Shape:  (2, 2)
Rank: tf.Tensor(2, shape=(), dtype=int32)
Size: tf.Tensor(4, shape=(), dtype=int32)

Tensor:
Shape:  (2, 2, 2)
Rank: tf.Tensor(3, shape=(), dtype=int32)
Size: tf.Tensor(8, shape=(), dtype=int32)
```

In [29]:

```
# @title 3. Create two tensors containing random values between 0 and 1 with shape [5, 30
0].

tensor1 = tf.random.uniform(shape=[5, 300])
tensor2 = tf.random.uniform(shape=[5, 300])

print("Tensor 1:", tensor1, "\n")
print("Tensor 2:", tensor2)
```

```
Tensor 1: tf.Tensor(
[[9.35069799e-01 2.93159366e-01 3.47624779e-01 ... 9.37980771e-01
  2.77746797e-01 9.39811587e-01]
 [3.13441038e-01 6.69866323e-01 8.31445694e-01 ... 5.01275063e-04
  8.27200532e-01 8.68953228e-01]
 [6.02149606e-01 2.56426811e-01 9.28914428e-01 ... 7.33514309e-01
  7.26268411e-01 6.11797929e-01]
 [1.14524126e-01 2.55079985e-01 1.60353541e-01 ... 8.12317371e-01
  1.08756304e-01 8.86413097e-01]
 [2.38683462e-01 5.10795832e-01 1.33285403e-01 ... 2.80616283e-02
  1.69330359e-01 4.04206514e-01]], shape=(5, 300), dtype=float32)

Tensor 2: tf.Tensor(
[[0.10613024 0.03080034 0.20274067 ... 0.5472406  0.94961333 0.47079873]
 [0.5672213  0.43569732 0.9779085  ... 0.31779325 0.9193777  0.8923398 ]
 [0.28223872 0.43564105 0.26612723 ... 0.43591607 0.8061352  0.01520658]
 [0.14729834 0.152084   0.04936051 ... 0.8428521  0.43748462 0.7309005 ]
 [0.39861667 0.1697557  0.7395948  ... 0.67870724 0.15673435 0.35786533]], shape=(5, 300)
, dtype=float32)
```

In [30]:

```
# @title 4. Multiply the two tensors you created in 3 using matrix multiplication.

tensor_mult = tf.matmul(tensor1, tf.transpose(tensor2))
print(tensor_mult)
```

```
tf.Tensor(
[[70.55658  74.646095 72.409355 73.96684  76.38922 ]
 [72.95719  73.301216 76.245316 76.13284  76.96216 ]
 [74.75396  73.05295  75.063736 76.718475 77.40128 ]
 [74.3958   72.595314 76.715935 78.66717  76.35787 ]
 [72.647026 71.64404  75.85521  75.28499  73.739876]], shape=(5, 5), dtype=float32)
```

In [31]:

```
# @title 5. Multiply the two tensors you created in 3 using dot product.

tensor_dot = tf.tensordot(tensor1, tf.transpose(tensor2), axes=1)
print(tensor_dot)
```

```
tf.Tensor(
[[70.55658  74.646095 72.409355 73.96684  76.38922 ]
 [72.95719  73.301216 76.245316 76.13284  76.96216 ]
 [74.75396  73.05295  75.063736 76.718475 77.40128 ]
 [74.3958   72.595314 76.715935 78.66717  76.35787 ]
 [72.647026 71.64404  75.85521  75.28499  73.739876]], shape=(5, 5), dtype=float32)
```

In [32]:

```
# @title 6. Create a tensor with random values between 0 and 1 with shape [224, 224, 3].

random_tensor = tf.random.uniform(shape=[224, 224, 3])
print(random_tensor)
```

```
tf.Tensor(
[[[7.24982142e-01 2.38647938e-01 4.72600341e-01]
  [6.83505535e-01 7.26200461e-01 5.55366516e-01]
  [2.76999474e-01 8.32804322e-01 2.70431638e-01]
  ...
  [2.11743236e-01 8.83840442e-01 9.72474813e-02]
  [6.87637925e-01 6.13827705e-01 2.86573887e-01]
  [3.06563377e-01 2.80949831e-01 4.17039394e-02]]

 [[8.65949750e-01 3.26418877e-02 9.29320335e-01]
  [5.11559248e-01 8.49012494e-01 8.89457345e-01]
  [7.79813528e-02 2.19149828e-01 9.52571511e-01]
  ...
  [6.48108244e-01 4.82172251e-01 4.43962812e-02]
  [3.58879089e-01 4.73972917e-01 3.34285259e-01]
  [8.73214602e-01 5.33488393e-01 2.79004574e-01]]

 [[7.68280029e-03 4.88049984e-01 5.21681309e-02]
  [9.02006030e-01 1.11773372e-01 3.88885736e-02]
  [1.73735142e-01 8.71878147e-01 8.38757873e-01]
  ...
  [6.38198376e-01 7.30853915e-01 6.33835793e-04]
  [2.44048238e-01 6.64864182e-01 6.12820148e-01]
  [3.92687321e-01 5.87354898e-02 6.38356447e-01]]

 ...

 [[5.06487846e-01 7.83785939e-01 2.03510046e-01]
  [2.08866596e-03 7.81955600e-01 9.61542606e-01]
  [1.49282932e-01 2.15043902e-01 7.42175937e-01]
  ...
  [6.62453532e-01 2.06840277e-01 9.41833138e-01]
  [6.88656807e-01 1.33257151e-01 7.61390924e-02]
  [5.88128090e-01 2.58996725e-01 4.24363971e-01]]

 [[9.84676957e-01 3.86990547e-01 2.79895902e-01]
  [3.18353534e-01 4.49264288e-01 2.37684488e-01]
  [7.76605844e-01 3.41882706e-01 1.01441145e-02]
  ...
  [7.70595074e-02 1.35366201e-01 6.46512389e-01]
  [4.16038871e-01 3.62759709e-01 2.25090027e-01]
  [3.82366538e-01 9.30075169e-01 3.69988799e-01]]

 [[1.50051713e-01 6.85753942e-01 5.81256032e-01]
  [8.70685101e-01 6.62837625e-01 9.97577071e-01]
  [5.52070141e-03 3.25226068e-01 6.93846822e-01]
  ...
  [2.31840730e-01 9.67443943e-01 1.29666090e-01]
  [8.67243052e-01 4.12011027e-01 8.59153390e-01]
  [5.77347636e-01 2.26977468e-01 7.91784048e-01]]], shape=(224, 224, 3), dtype=float32)
```

In [33]:

```
# @title 7. Find the min and max values of the tensor you created in 6.

print("Minimum:", tf.reduce_min(random_tensor))
print("Maximum:", tf.reduce_max(random_tensor))
```

```
Minimum: tf.Tensor(8.34465e-06, shape=(), dtype=float32)
Maximum: tf.Tensor(0.9999969, shape=(), dtype=float32)
```

In [34]:

```
# @title 8. Created a tensor with random values of shape [1, 224, 224, 3] then squeeze it
to change the shape to [224, 224, 3].

# Create a tensor with random values of shape [1, 224, 224, 3]
random_tensor_squeezable = tf.random.Generator.from_seed(42)
random_tensor_squeezable = random_tensor_squeezable.normal(shape=(1, 224, 224, 3))

# Squeeze the tensor to change the shape to [224, 224, 3]
squeezed_tensor = tf.squeeze(random_tensor_squeezable)

# Print the shapes of the original and squeezed tensors
print("Original tensor shape:", random_tensor_squeezable.shape)
print("Squeezed tensor shape:", squeezed_tensor.shape)
```

```
Original tensor shape: (1, 224, 224, 3)
Squeezed tensor shape: (224, 224, 3)
```

In [35]:

```
# @title 9. Create a tensor with shape [10] using your own choice of values, then find th
e index which has the maximum value.

# Create a tensor with shape [10]
my_tensor = tf.constant([5, 4, 3, 2, 1, 9, 8, 7, 6, 0])

# Find the index with the maximum value
max_index = tf.argmax(my_tensor)

print("Tensor:", my_tensor)
print("Index with maximum value:", max_index.numpy())
```

```
Tensor: tf.Tensor([5 4 3 2 1 9 8 7 6 0], shape=(10,), dtype=int32)
Index with maximum value: 5
```

In [36]:

```
# @title 10. One-hot encode the tensor you created in 9.

one_hot_tensor = tf.one_hot(my_tensor, depth=tf.size(my_tensor).numpy())

print("One-hot encoded tensor:\n", one_hot_tensor)
```

```
One-hot encoded tensor:
 tf.Tensor(
[[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]], shape=(10, 10), dtype=float32)
```