```
import tensorflow as tf
print(tf.__version__)
```

```
2.17.1
```

```
scalar=tf.constant(7)
scalar
```

```
<tf.Tensor: shape=(), dtype=int32, numpy=7>
```

tf.constant()

```
#check no of dimentions
scalar.ndim
```

```
0
```

```
#create a vector
vector=tf.constant([[10,10],[10,10]])
vector
```

```
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[10, 10],
       [10, 10]], dtype=int32)>
```

```
vector.ndim
```

```
2
```

```
matrix=tf.constant([[10.,7.],[7.,10.],[3.,2.]],dtype=tf.float16)
matrix
```

```
<tf.Tensor: shape=(3, 2), dtype=float16, numpy=
array([[10.,  7.],
       [ 7., 10.],
       [ 3.,  2.]], dtype=float16)>
```

```
#create tensor
tensor =tf.constant([[[1,2,3,],[3,4,5]],[[6,8,9],[10,11,12]]])
tensor
```

```
<tf.Tensor: shape=(2, 2, 3), dtype=int32, numpy=
array([[[ 1,  2,  3],
        [ 3,  4,  5]],

       [[ 6,  8,  9],
        [10, 11, 12]]], dtype=int32)>
```

```
tensor.ndim
```

```
3
```

tf.Variable()

```
changable_tensor=tf.Variable([10,7])
unchangable_tensor=tf.constant([10,7])
changable_tensor,unchangable_tensor
```

```
(<tf.Variable 'Variable:0' shape=(2,) dtype=int32, numpy=array([10,  7],
dtype=int32)>,
 <tf.Tensor: shape=(2,), dtype=int32, numpy=array([10,  7], dtype=int32)>)
```

```
#.assign()
changable_tensor[0].assign(7)
changable_tensor
```

```
<tf.Variable 'Variable:0' shape=(2,) dtype=int32, numpy=array([7, 7], dtype=int32)>
```

```
#unchangable_tensor[0].assign(7)
#unchangable_tensor
```

```
#create random tensor
r1=tf.random.Generator.from_seed(42)
r1=r1.normal(shape=(3,2))
r1
```

```
<tf.Tensor: shape=(3, 2), dtype=float32, numpy=
array([[-0.7565803 , -0.06854702],
       [ 0.07595026, -1.2573844 ],
       [-0.23193763, -1.8107855 ]], dtype=float32)>
```

shuffle the order of elements

```
tf.random.set_seed(42)#global level seed
tf.random.shuffle(r1,seed=42)#operation level seed
```

```
<tf.Tensor: shape=(3, 2), dtype=float32, numpy=
array([[-0.7565803 , -0.06854702],
       [ 0.07595026, -1.2573844 ],
       [-0.23193763, -1.8107855 ]], dtype=float32)>
```

other ways to make tensors

```
tf.ones([10,7])#tensor of all ones
```

```
<tf.Tensor: shape=(10, 7), dtype=float32, numpy=
array([[1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.]], dtype=float32)>
```

```
tf.zeros(shape=(3,4))
```

```
<tf.Tensor: shape=(3, 4), dtype=float32, numpy=
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]], dtype=float32)>
```

main diff b/w numpy arrays and tensor flow tensors is that tensors can be run on a gpu much faster

```
import numpy as np
numpy_A=np.arange(1,25,dtype=np.int64)
numpy_A
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24])
```

```
A=tf.constant(numpy_A,shape=(2,3,4))
B=tf.constant(numpy_A)
A,B
```

```
(<tf.Tensor: shape=(2, 3, 4), dtype=int64, numpy=
 array([[[ 1,  2,  3,  4],
         [ 5,  6,  7,  8],
         [ 9, 10, 11, 12]],

        [[13, 14, 15, 16],
         [17, 18, 19, 20],
         [21, 22, 23, 24]]])>,
 <tf.Tensor: shape=(24,), dtype=int64, numpy=
 array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
        18, 19, 20, 21, 22, 23, 24])>)
```

getting info from tensors shape(len of each of the dim)tensor.shape ,rank(the no of tenssor dim)tensor.ndim ,axis(a particular dim of a tansor)tensor[0] ,size(total no of items in the tansor)tf.size(tensor)

```
rank_4_tensor=tf.zeros(shape=(2,3,4,5))
rank_4_tensor
```

```
<tf.Tensor: shape=(2, 3, 4, 5), dtype=float32, numpy=
array([[[[0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.]],

        [[0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.]],

        [[0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.]]],


       [[[0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.]],

        [[0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.]],

        [[0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0.]]]], dtype=float32)>
```

```
rank_4_tensor.ndim,rank_4_tensor.shape,tf.size(rank_4_tensor)
```

```
(4, TensorShape([2, 3, 4, 5]), <tf.Tensor: shape=(), dtype=int32, numpy=120>)
```

Suggested code may be subject to a licence | | Justy-11/DL_ML_fundamentals

```
print("data type of elements:",rank_4_tensor.dtype)
print("no of dimensions(rank):",rank_4_tensor.ndim)
print("shape of tensor:",rank_4_tensor.shape)
print("elements along the 0 axis:",rank_4_tensor.shape[0])
print("elements along the last axis:",rank_4_tensor.shape[-1])
print("total no of elements in our tensor:",tf.size(rank_4_tensor).numpy())
```

```
data type of elements: <dtype: 'float32'>
no of dimensions(rank): 4
shape of tensor: (2, 3, 4, 5)
elements along the 0 axis: 2
elements along the last axis: 5
total no of elements in our tensor: 120
```

indexing

```
#get first two elements of each dimention
rank_4_tensor[:2,:2,:2,:2]
```

```
<tf.Tensor: shape=(2, 2, 2, 2), dtype=float32, numpy=
array([[[[0., 0.],
         [0., 0.]],

        [[0., 0.],
         [0., 0.]]],


       [[[0., 0.],
         [0., 0.]],

        [[0., 0.],
         [0., 0.]]]], dtype=float32)>
```

```
#get first element from each dim except for final one
rank_4_tensor[:1,:1,:1]
```

```
<tf.Tensor: shape=(1, 1, 1, 5), dtype=float32, numpy=array([[[[0., 0., 0., 0.,
0.]]]], dtype=float32)>
```

```
rank_2_tensor=tf.constant([[10,7],[3,4]])
rank_2_tensor.shape,rank_2_tensor.ndim
```

```
(TensorShape([2, 2]), 2)
```

```
rank_2_tensor[:,-1]
```

```
<tf.Tensor: shape=(2,), dtype=int32, numpy=array([7, 4], dtype=int32)>
```

Suggested code may be subject to a licence | bhupendpatil/Practice

```
#add extra dimention
rank_3_tensor=rank_2_tensor[...,tf.newaxis]
rank_3_tensor
```

```
<tf.Tensor: shape=(2, 2, 1), dtype=int32, numpy=
array([[[10],
        [ 7]],

       [[ 3],
        [ 4]]], dtype=int32)>
```

```
#alternative to tf.newaxis
tf.expand_dims(rank_2_tensor,axis=-1)
```

```
<tf.Tensor: shape=(2, 2, 1), dtype=int32, numpy=
array([[[10],
        [ 7]],

       [[ 3],
        [ 4]]], dtype=int32)>
```

```
tf.expand_dims(rank_2_tensor,axis=0)
```

```
<tf.Tensor: shape=(1, 2, 2), dtype=int32, numpy=
array([[[10,  7],
        [ 3,  4]]], dtype=int32)>
```

Manipulating tensors

```
#add values to a tensor using addition operator
tensor=tf.constant([[10,7],[3,4]])
tensor+10
```

```
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[20, 17],
       [13, 14]], dtype=int32)>
```

```
tensor*10#works
tensor-10
```

```
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[ 0, -3],
       [-7, -6]], dtype=int32)>
```

```
tf.multiply(tensor,10)#use this
```

```
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[100,  70],
       [ 30,  40]], dtype=int32)>
```

Mtrix multiplication

```
tf.matmul(tensor,tensor)
```

```
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[121,  98],
       [ 42,  37]], dtype=int32)>
```

```
#matrix mul usin @
tensor @ tensor
```

```
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[121,  98],
       [ 42,  37]], dtype=int32)>
```

can use tf.reshape(y,shape=(2,3))

```
x= tf.constant([[3,4],[5,6],[7,8]])
y= tf.constant([[1,2],[3,4],[7,5]])
```

```
#transpose
```

```
tf.transpose(x),tf.reshape(x,shape=(2,3))
```

```
(<tf.Tensor: shape=(2, 3), dtype=int32, numpy=
 array([[3, 5, 7],
        [4, 6, 8]], dtype=int32)>,
 <tf.Tensor: shape=(2, 3), dtype=int32, numpy=
 array([[3, 4, 5],
        [6, 7, 8]], dtype=int32)>)
```

matrix multiplication: tf.matmul(), tf.tensordot(),@

```
tf.matmul(x,tf.reshape(y,shape=(2,3)))
```

```
<tf.Tensor: shape=(3, 3), dtype=int32, numpy=
array([[19, 34, 29],
       [29, 52, 45],
       [39, 70, 61]], dtype=int32)>
```

```
tf.matmul(x,tf.transpose(y))
```

```
<tf.Tensor: shape=(3, 3), dtype=int32, numpy=
array([[11, 25, 41],
       [17, 39, 65],
       [23, 53, 89]], dtype=int32)>
```

```
transpose_y=tf.transpose(y)
reshape_y=tf.reshape(y,shape=(2,3))
transpose_y,reshape_y
```

```
(<tf.Tensor: shape=(2, 3), dtype=int32, numpy=
 array([[1, 3, 7],
        [2, 4, 5]], dtype=int32)>,
 <tf.Tensor: shape=(2, 3), dtype=int32, numpy=
 array([[1, 2, 3],
        [4, 7, 5]], dtype=int32)>)
```

Changing the datatype of a tensor

```
B = tf.constant([[2.4,5.6],[1.2,2.3]])
B.dtype
```

```
tf.float32
```

```
A= tf.constant([[1,2],[2,3]])
A.dtype
```

```
tf.int32
```

```
D = tf.cast(B,dtype=tf.float16)
D
```

```
<tf.Tensor: shape=(2, 2), dtype=float16, numpy=
array([[2.4, 5.6],
       [1.2, 2.3]], dtype=float16)>
```

AGGREGATING TENSORS

```
F= tf.constant([[1,2],[-3,-4]])
```

```
tf.abs(F)
```

```
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[1, 2],
       [3, 4]], dtype=int32)>
```

min ,max, mean ,sum

```
 # This is formatted as code
```

```
E= tf.constant(np.random.randint(0,100,size=50))
E
```

```
<tf.Tensor: shape=(50,), dtype=int64, numpy=
array([69, 28, 69, 80,  8, 53, 35, 62, 26, 68, 95, 50, 25, 92, 85, 38, 21,
       44, 72, 14, 78, 91,  2, 97, 27, 13, 10, 72, 26, 10,  0, 94, 75, 14,
       44, 42, 94, 85, 41, 96, 95, 77, 56, 67, 36, 87, 26, 98,  8, 42])>
```

```
#min
```

```
tf.reduce_min(E).numpy()
```

```
0
```

```
tf.reduce_max(E)
```

```
<tf.Tensor: shape=(), dtype=int64, numpy=98>
```

```
tf.reduce_mean(E)
```

```
<tf.Tensor: shape=(), dtype=int64, numpy=52>
```

```
tf.reduce_sum(E)
```

```
<tf.Tensor: shape=(), dtype=int64, numpy=2637>
```

```
std_dev = tf.math.reduce_std(tf.cast(E,dtype=tf.float32))#should be in float
std_dev
```

```
<tf.Tensor: shape=(), dtype=float32, numpy=30.827137>
```

tensorflow probability as tfp

```
import tensorflow_probability as tfp
tfp.stats.variance(E),tf.math.reduce_variance(tf.cast(E,dtype=tf.float32))
```

```
(<tf.Tensor: shape=(), dtype=int64, numpy=950>,
 <tf.Tensor: shape=(), dtype=float32, numpy=950.3123>)
```

FIND THE POSITIONAL MAX AND MIN

```
F = tf.random.uniform(shape=[50])
F
```

```
<tf.Tensor: shape=(50,), dtype=float32, numpy=
array([0.6645621 , 0.44100678, 0.3528825 , 0.46448255, 0.03366041,
       0.68467236, 0.74011743, 0.8724445 , 0.22632635, 0.22319686,
       0.3103881 , 0.7223358 , 0.13318717, 0.5480639 , 0.5746088 ,
       0.8996835 , 0.00946367, 0.5212307 , 0.6345445 , 0.1993283 ,
       0.72942245, 0.54583454, 0.10756552, 0.6767061 , 0.6602763 ,
       0.33695042, 0.60141766, 0.21062577, 0.8527372 , 0.44062173,
       0.9485276 , 0.23752594, 0.81179297, 0.5263394 , 0.494308  ,
       0.21612847, 0.8457197 , 0.8718841 , 0.3083862 , 0.6868038 ,
       0.23764038, 0.7817228 , 0.9671384 , 0.06870162, 0.79873943,
       0.66028714, 0.5871513 , 0.16461694, 0.7381023 , 0.32054043],
      dtype=float32)>
```

```
tf.argmax(F)#positional max
```

```
<tf.Tensor: shape=(), dtype=int64, numpy=42>
```

```
#index on our largest value position
F[tf.argmax(F)]
```

```
<tf.Tensor: shape=(), dtype=float32, numpy=0.9671384>
```

```
tf.reduce_max(F)
```

```
<tf.Tensor: shape=(), dtype=float32, numpy=0.9671384>
```

SQUEEZING A TENSOR (REMOVING ALL SINGLE DIMENTIONS)

```
G = tf.constant(tf.random.uniform(shape=[50]),shape=(1,1,1,1,50))
G
```

```
<tf.Tensor: shape=(1, 1, 1, 1, 50), dtype=float32, numpy=
array([[[[[0.7413678 , 0.62854624, 0.01738465, 0.3431449 , 0.51063764,
           0.3777541 , 0.07321596, 0.02137029, 0.2871771 , 0.4710616 ,
                0.6936141 , 0.07321334, 0.93251204, 0.20843053, 0.70105827,
                0.45856392, 0.8596262 , 0.92934334, 0.20291913, 0.76865506,
                0.60016024, 0.27039742, 0.88180614, 0.05365038, 0.42274463,
                0.89037776, 0.7887033 , 0.10165584, 0.19408834, 0.27896714,
                0.39512634, 0.12235212, 0.38412368, 0.9455296 , 0.77594674,
                0.94442344, 0.04296565, 0.4746096 , 0.6548251 , 0.5657116 ,
                0.13858628, 0.3004663 , 0.3311677 , 0.12907016, 0.6435652 ,
                0.45473957, 0.68881893, 0.30203617, 0.49152803, 0.26529062]]]]],
           dtype=float32)>
```

```
G_squeezed = tf.squeeze(G)
G_squeezed,G_squeezed.shape#reduce single dim
```

```
(<tf.Tensor: shape=(50,), dtype=float32, numpy=
 array([0.7413678 , 0.62854624, 0.01738465, 0.3431449 , 0.51063764,
        0.3777541 , 0.07321596, 0.02137029, 0.2871771 , 0.4710616 ,
        0.6936141 , 0.07321334, 0.93251204, 0.20843053, 0.70105827,
        0.45856392, 0.8596262 , 0.92934334, 0.20291913, 0.76865506,
        0.60016024, 0.27039742, 0.88180614, 0.05365038, 0.42274463,
        0.89037776, 0.7887033 , 0.10165584, 0.19408834, 0.27896714,
        0.39512634, 0.12235212, 0.38412368, 0.9455296 , 0.77594674,
        0.94442344, 0.04296565, 0.4746096 , 0.6548251 , 0.5657116 ,
        0.13858628, 0.3004663 , 0.3311677 , 0.12907016, 0.6435652 ,
        0.45473957, 0.68881893, 0.30203617, 0.49152803, 0.26529062],
       dtype=float32)>,
 TensorShape([50]))
```

ONE HOT ENCODING

```
some_list=[0,1,2,3]# red,green,blue,purple
#one hot encode
tf.one_hot(some_list,depth=4)
```

```
<tf.Tensor: shape=(4, 4), dtype=float32, numpy=
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]], dtype=float32)>
```

```
#squaring,log,sqrt
```

```
H= tf.range(1,10)
H
```

```
<tf.Tensor: shape=(9,), dtype=int32, numpy=array([1, 2, 3, 4, 5, 6, 7, 8, 9],
dtype=int32)>
```

```
tf.square(H)
```

```
<tf.Tensor: shape=(9,), dtype=int32, numpy=array([ 1,  4,  9, 16, 25, 36, 49, 64,
81], dtype=int32)>
```

```
tf.sqrt(tf.cast(H,dtype=tf.float32))
```

```
<tf.Tensor: shape=(9,), dtype=float32, numpy=
array([1.       , 1.4142135, 1.7320508, 2.       , 2.236068 , 2.4494898,
       2.6457512, 2.828427 , 3.       ], dtype=float32)>
```

```
tf.math.log(tf.cast(H,dtype=tf.float32))
```

```
<tf.Tensor: shape=(9,), dtype=float32, numpy=
array([0.       , 0.6931472, 1.0986123, 1.3862944, 1.609438 , 1.7917595,
       1.9459102, 2.0794415, 2.1972246], dtype=float32)>
```

TENSORS AND NUMPY(COMPATIBLE)

```
J =tf.constant(np.array([3.,5.,6.]))
J
```

```
<tf.Tensor: shape=(3,), dtype=float64, numpy=array([3., 5., 6.])>
```

```
np.array(J),J.numpy()
```

```
(array([3., 5., 6.]), array([3., 5., 6.]))
```

ACCESS TO GPU'S

```
import tensorflow as tf
tf.config.list_physical_devices("GPU")#no access
```

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

```
!nvidia-smi
```

```
Fri Nov 29 11:23:18 2024
+-----------------------------------------------------------------------------------
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2
|-----------------------------------------+----------------------+-------------------
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. E
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute
|                                         |                      |               MIG
|=========================================+======================+===================
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |
| N/A   54C    P8              10W /  70W |      3MiB / 15360MiB |      0%      Defau
|                                         |                      |                  N
+-----------------------------------------+----------------------+-------------------

+-----------------------------------------------------------------------------------
| Processes:
|  GPU   GI   CI        PID   Type   Process name                            GPU Memo
|        ID   ID                                                             Usage
|===================================================================================
|  No running processes found
```

```
+------------------------------------------------------------------------------------
```

```
Start coding or generate with AI.
```