```
import tensorflow as tf
```

Slicing and indexing in tensorfow

```
vector = tf.constant([1,2,3,4,5,6,7,8,9])
vector
```

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

```
vector[0]
```

```
<tf.Tensor: shape=(), dtype=int32, numpy=1>
```

```
vector[1:-1]
```

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

```
vector[:]
```

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

```
vector[-1:]
```

```
<tf.Tensor: shape=(1,), dtype=int32, numpy=array([9], dtype=int32)>
```

```
vector[1:5]
```

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

```
vector[::-1]
```

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

```
# fetch mutiple element from tensor for multiple indices: tf.gather()
indices = tf.constant([4,8])
tf.gather(vector,indices)
```

```
<tf.Tensor: shape=(2,), dtype=int32, numpy=array([5, 9], dtype=int32)>
```

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

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

```
# fetch all element from row 0
metrix[0,:]
```

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

```
# fetch all element from all row
metrix[:,:]
```

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

```
# fetch first two elements from all rows
metrix[:,:2]
```

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

```
metrix[:2,:]
```

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

range function in tensorflow

syntax tf.range(start, end, delta=1, dtype=None, name='range')

```
rangetensor = tf.range(start=5,limit=15)
rangetensor
```

```
<tf.Tensor: shape=(10,), dtype=int32, numpy=array([ 5,  6,  7,  8,  9, 10, 11, 12, 13,
14], dtype=int32)>
```

```
limit=10
tf.range(limit)
```

```
<tf.Tensor: shape=(10,), dtype=int32, numpy=array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
dtype=int32)>
```

```
odd_number = tf.range(start=2,limit=21,delta=2)
even_number = tf.range(start=1,limit=20,delta=2)
print("Odd number : ",odd_number)
print("Even number :",even_number)
```

```
Odd number :  tf.Tensor([ 2  4  6  8 10 12 14 16 18 20], shape=(10,), dtype=int32)
Even number : tf.Tensor([ 1  3  5  7  9 11 13 15 17 19], shape=(10,), dtype=int32)
```

```
reverse_number = tf.range(start=20,limit=0,delta=-1)
reverse_number
```

```
<tf.Tensor: shape=(20,), dtype=int32, numpy=
array([20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10,  9,  8,  7,  6,  5,  4,
        3,  2,  1], dtype=int32)>
```

type conversion in tensorflow

```
tensor_float = tf.constant([2.,3.,4.])
tensor_float, tensor_float.dtype
```

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

```
tensor_int = tf.constant([2,3,4])
tensor_int,tensor_int.dtype
```

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

```
tf.cast(tensor_int,dtype=tf.float32)
```

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

```
tf.cast(tensor_float,dtype=tf.int32)
```

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

```
tf.cast(tensor_float,dtype=tf.float16)
```

```
<tf.Tensor: shape=(3,), dtype=float16, numpy=array([2., 3., 4.], dtype=float16)>
```

```
print("real tensor_int value :",tensor_int)
update = tf.cast(tensor_int,dtype=tf.int16)
print("changing it into int16 type :",update)
```

```
real tensor_int value : tf.Tensor([2 3 4], shape=(3,), dtype=int32)
changing it into int16 type : tf.Tensor([2 3 4], shape=(3,), dtype=int16)
```

```
tf.constant(['a','b','c'])
```

```
<tf.Tensor: shape=(3,), dtype=string, numpy=array([b'a', b'b', b'c'], dtype=object)>
```

Basic Math Operation in tensorflow

```
x = tf.constant([10.,20.,30.])
y = tf.constant([1.,2.,3.])
x,y
```

```
(<tf.Tensor: shape=(3,), dtype=float32, numpy=array([10., 20., 30.], dtype=float32)>,
 <tf.Tensor: shape=(3,), dtype=float32, numpy=array([1., 2., 3.], dtype=float32)>)
```

```
x+10
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([20., 30., 40.], dtype=float32)>
```

```
x+y
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([11., 22., 33.], dtype=float32)>
```

```
add = tf.add(x,y)
add
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([11., 22., 33.], dtype=float32)>
```

```
x-10
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([ 0., 10., 20.], dtype=float32)>
```

```
x-y
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([ 9., 18., 27.], dtype=float32)>
```

```
sub = tf.subtract(x,y)
sub
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([ 9., 18., 27.], dtype=float32)>
```

```
x*10
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([100., 200., 300.], dtype=float32)>
```

```
x*y
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([10., 40., 90.], dtype=float32)>
```

```
mul = tf.multiply(x,y)
mul
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([10., 40., 90.], dtype=float32)>
```

```
x/10
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([1., 2., 3.], dtype=float32)>
```

```
x/y
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([10., 10., 10.], dtype=float32)>
```

```
div = tf.divide(x,y)
div
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([10., 10., 10.], dtype=float32)>
```

```
square = tf.square(5)
square
```

```
<tf.Tensor: shape=(), dtype=int32, numpy=25>
```

```
squareofx = tf.square(x)
squareofx
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([100., 400., 900.], dtype=float32)>
```

```
x*x
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([100., 400., 900.], dtype=float32)>
```

```
power = tf.pow(x,3)
power
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([ 1000.,  8000., 27000.],
dtype=float32)>
```

```
x**3
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([ 1000.,  8000., 27000.],
dtype=float32)>
```

```
sq_diff = tf.math.squared_difference(x,y)
sq_diff
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([ 81., 324., 729.], dtype=float32)>
```

```
(x-y) ** 2
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([ 81., 324., 729.], dtype=float32)>
```

```
tensor = tf.constant([30,10,70,60,20,90,80,40,50])
tensor
```

```
<tf.Tensor: shape=(9,), dtype=int32, numpy=array([30, 10, 70, 60, 20, 90, 80, 40, 50],
dtype=int32)>
```

```
min_element_val = tf.argmin(tensor)
max_element_val = tf.argmax(tensor)
print("min tensor value : ",min_element_val)
print("max tensor value : ",max_element_val)
```

```
min tensor value :  tf.Tensor(1, shape=(), dtype=int64)
max tensor value :  tf.Tensor(5, shape=(), dtype=int64)
```

```
tensor[min_element_val].numpy(), tensor[max_element_val].numpy()
```

```
(10, 90)
```

```
min = tf.reduce_min(tensor)
max = tf.reduce_max(tensor)
min,max
```

```
(<tf.Tensor: shape=(), dtype=int32, numpy=10>,
 <tf.Tensor: shape=(), dtype=int32, numpy=90>)
```

```
matrix = tf.constant([[10,20,30],[40,50,60]])
matrix
```

```
<tf.Tensor: shape=(2, 3), dtype=int32, numpy=
array([[10, 20, 30],
       [40, 50, 60]], dtype=int32)>
```

```
sum =tf.reduce_sum(matrix)
sum
```

```
<tf.Tensor: shape=(), dtype=int32, numpy=210>
```

```
# to find sunm in row wise
sum_0 = tf.reduce_sum(matrix,axis=0)

# to find sum in column wise
sum_1 = tf.reduce_sum(matrix,axis=1)

sum_0.numpy(), sum_1.numpy()
```

```
(array([50, 70, 90], dtype=int32), array([ 60, 150], dtype=int32))
```

```
# find mean of tensor
mean = tf.reduce_mean(matrix)
mean
```

```
<tf.Tensor: shape=(), dtype=int32, numpy=35>
```

find variance of a tensor

```
import tensorflow_probability as tfp
```

```
variance = tfp.stats.variance(tensor)
variance
```

```
<tf.Tensor: shape=(), dtype=int32, numpy=666>
```

dot product

```
x,y
```

```
(<tf.Tensor: shape=(3,), dtype=float32, numpy=array([10., 20., 30.], dtype=float32)>,
 <tf.Tensor: shape=(3,), dtype=float32, numpy=array([1., 2., 3.], dtype=float32)>)
```

```
# getting dot product without tensor method
dot_product_1 = tf.reduce_sum(tf.multiply(x,y))
dot_product_1
```

```
<tf.Tensor: shape=(), dtype=float32, numpy=140.0>
```

```
tf.tensordot(x,y,axes=1)
```

```
<tf.Tensor: shape=(), dtype=float32, numpy=140.0>
```

```
z = tf.constant([9.,16.,25.])
z
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([ 9., 16., 25.], dtype=float32)>
```

```
sqrt = tf.sqrt(z)
sqrt
```

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

metrix

```
metrix11 = tf.constant([[1,2,3],[3,4,5]])
metrix21 = tf.constant([[1,2],[3,4]])
metrix31 = tf.constant([[10,20,30],[30,40,50]])
metrix11,metrix21,metrix31
```

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

```
       [3, 4, 5]], dtype=int32)>,
 <tf.Tensor: shape=(2, 2), dtype=int32, numpy=
 array([[1, 2],
        [3, 4]], dtype=int32)>,
 <tf.Tensor: shape=(2, 3), dtype=int32, numpy=
 array([[10, 20, 30],
        [30, 40, 50]], dtype=int32)>)
```

```
matmull_2 = tf.matmul(metrix11 ,metrix21 )
matmull_2
```

```
---------------------------------------------------------------------------
InvalidArgumentError                      Traceback (most recent call last)
<ipython-input-90-ec85895a0314> in <cell line: 1>()
----> 1 matmull_2 = tf.matmul(metrix11 ,metrix21 )
      2 matmull_2

 1 frames 
/usr/local/lib/python3.10/dist-packages/tensorflow/python/framework/ops.py in
raise_from_not_ok_status(e, name)
   6654 def raise_from_not_ok_status(e, name):
   6655   e.message += (" name: " + str(name if name is not None else ""))
-> 6656   raise core._status_to_exception(e) from None  # pylint: disable=protected-
access
   6657
   6658

InvalidArgumentError: {{function_node
__wrapped__MatMul_device_/job:localhost/replica:0/task:0/device:CPU:0}} Matrix size-
incompatible: In[0]: [2,3], In[1]: [2,2] [Op:MatMul] name:

SEARCH STACK OVERFLOW
```

```
metrix11 @ metrix31
```

```
---------------------------------------------------------------------------
InvalidArgumentError                      Traceback (most recent call last)
<ipython-input-91-ac9a09095f5d> in <cell line: 1>()
----> 1 metrix11 @ metrix31
```

```
metrix11.shape
```

```
TensorShape([2, 3])
```

```
  6654 def raise_from_not_ok_status(e, name):
```

```
metrixreshaped = tf.reshape(metrix11,shape=(3,2))
metrixreshaped
```

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

```
Incompatible: In[0]: [2,3], In[1]: [2,3] [Op:MatMul] name:
```

```
metrix11
```

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

```
matrix_transpost = tf.transpose(metrix11)
matrix_transpost
```

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

```
zeros = tf.zeros(shape=(5,5))
zeros
```

```
<tf.Tensor: shape=(5, 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.]], dtype=float32)>
```

```
metrix11.shape
```

```
TensorShape([2, 3])
```

```
tf.zeros_like(metrix11)
```

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

```
       [0, 0, 0]], dtype=int32)>
```

```
ones = tf.ones(shape=(5,5))
ones
```

```
<tf.Tensor: shape=(5, 5), 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.]], dtype=float32)>
```

```
tf.ones_like(metrix11)
```

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

```
identity = tf.eye(4)
identity
```

```
<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)>
```

```
mt1 = [[3,5,7],[7,7,7]]
mt2 = [[4,5,6],[5,5,5]]
mt1,mt2
```

```
([[3, 5, 7], [7, 7, 7]], [[4, 5, 6], [5, 5, 5]])
```

```
tf.concat([mt1,mt2],axis=0)
```

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

```
tf.concat([mt1,mt2],axis=1)
```

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

```
tf.stack([mt1,mt2],axis=0)
```

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

       [[4, 5, 6],
        [5, 5, 5]]], dtype=int32)>
```

```
tf.stack([mt1,mt2],axis=1)
```

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

       [[7, 7, 7],
        [5, 5, 5]]], dtype=int32)>
```

```
tf.stack([mt1,mt2],axis=1).ndim
```

```
3
```