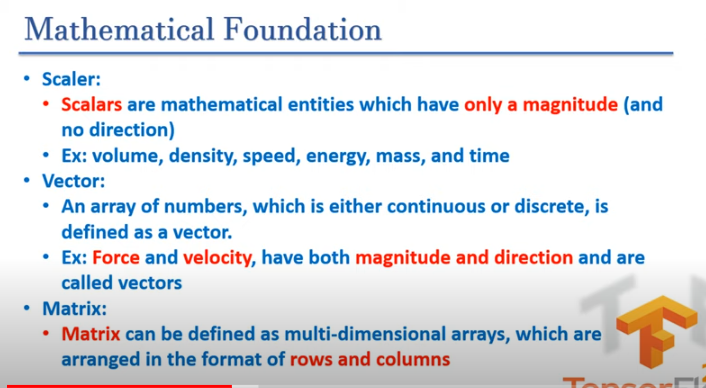
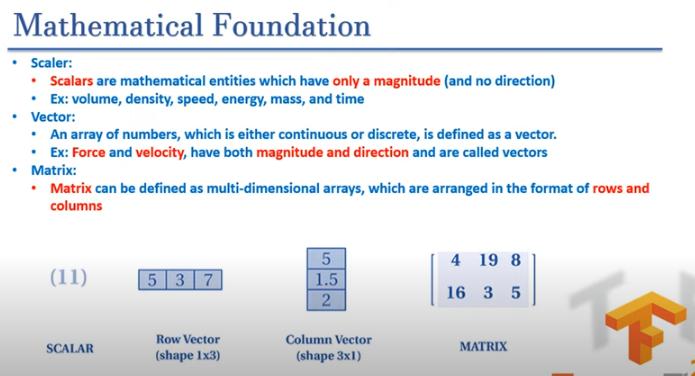
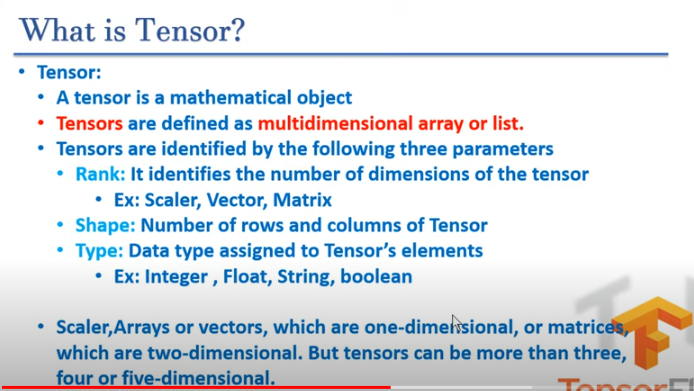
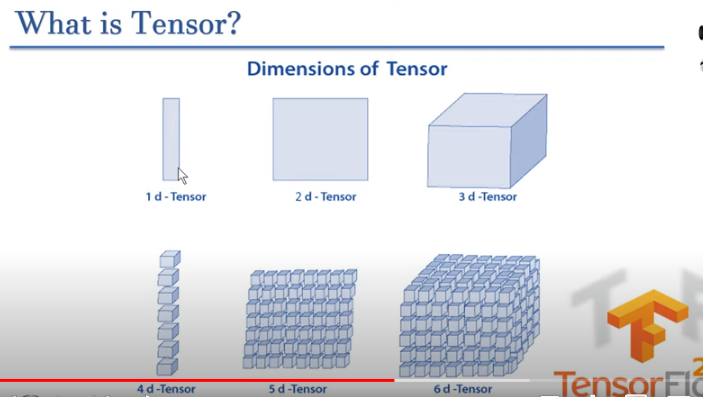
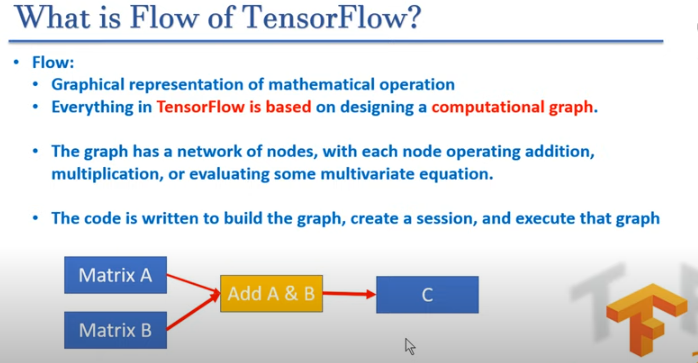
[**https://www.tensorflow.org/**](https://www.tensorflow.org/)

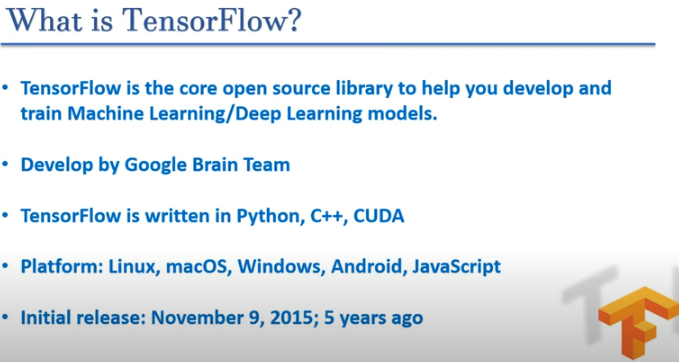


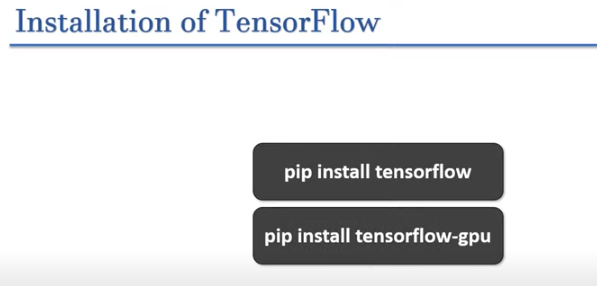


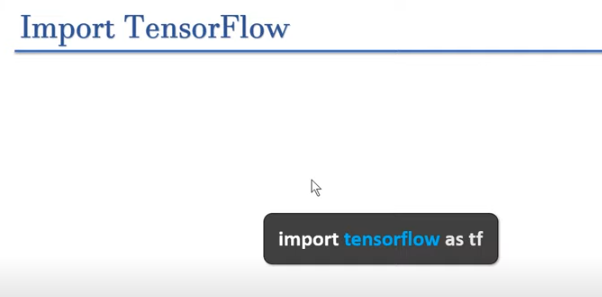


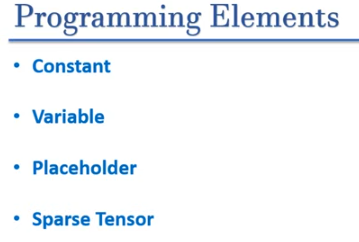


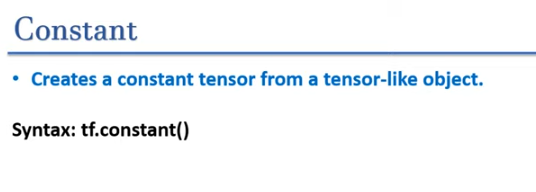


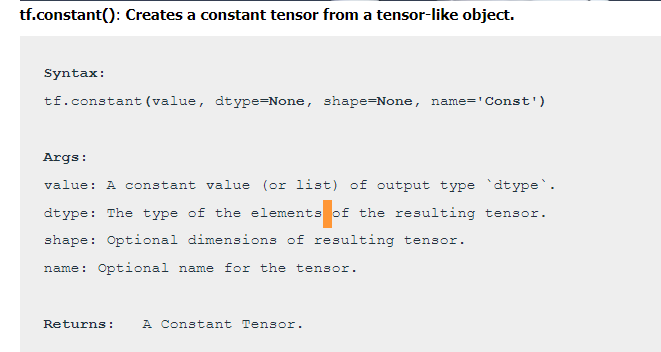


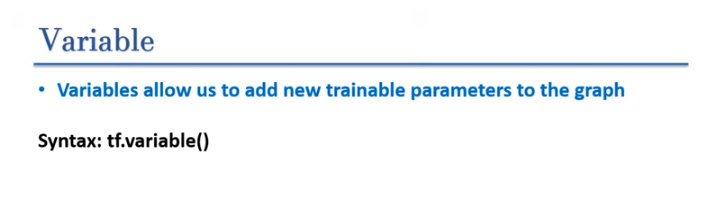


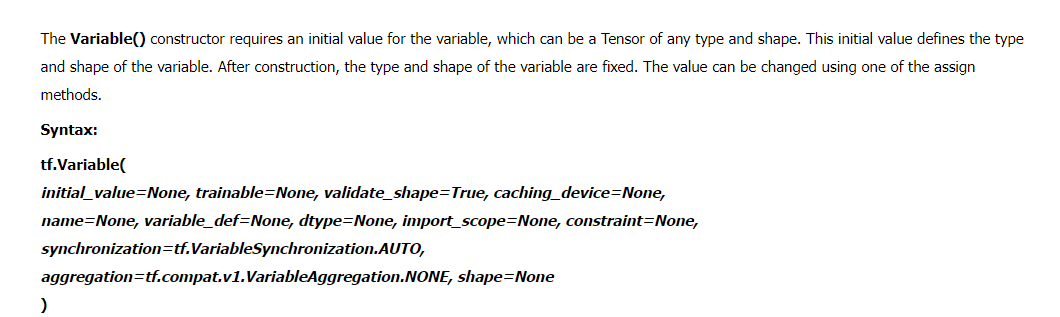














A **TensorFlow placeholder**is simply a variable that we will assign data to at a later date. It allows us to create our operations and build our computation graph, without needing the data.

***Syntax: tf.compat.v1.placeholder(dtype, shape=None, name=None)***

***Note:***

***TensorFlow placeholder is not available in TensorFlow 2.x.***

***In TensorFlow 1.x, tf.placeholder was used to create a tensor that is used for feeding input data into the computation graph. However, TensorFlow 2.x has deprecated the use of placeholders as it shifted to an eager execution mode by default, which means computations are executed immediately as they are called within Python. This approach eliminates the need for placeholders.***

***Alternatives to tf.placeholder in TensorFlow 2.x:***

* ***tf.Variable: Used for trainable variables.***
* ***tf.constant: Used for constants that do not change.***
* ***tf.Tensor: For directly creating tensor objects from data, since TensorFlow 2.x operates in eager execution mode.***



**What is Sparse Tensor?**

***Tensor that contain mostly zero values are called sparse tensor.***

When working with tensors that contain a lot of zero values, it is important to store them in a space- and time-efficient manner. Sparse tensors enable efficient storage and processing of tensors that contain a lot of zero values.  
Sparse tensors are used extensively in encoding schemes like TF-IDF as part of data pre-processing in NLP applications and for pre-processing images with a lot of dark pixels in computer vision applications.

Currently, sparse tensors in TensorFlow are encoded using the coordinate list (COO) format.

The COO encoding for sparse tensors is comprised of:

values: A 1D tensor with shape [N] containing all nonzero values.  
indices: A 2D tensor with shape [N, rank], containing the indices of the nonzero values.  
dense\_shape: A 1D tensor with shape [rank], specifying the shape of the tensor.

**A nonzero value in the context of a tf.SparseTensor is a value that’s not explicitly encoded.**

**Syntax:** **tf.sparse.SparseTensor(*indices, values, dense\_shape*)**

**Create Ones vs Ones\_like vs Ones\_initializer Tensor Using TensorFlow 2.0 :**

**1. tf.ones()**

The tf.ones() function is used to create a new tensor with all elements set to one, and you can specify the shape and data type of the tensor.

**Syntax:  
tf.ones(shape, dtype=tf.dtypes.float32, name=None) -> Return Ones Tensor**

**2. tf.ones\_like()**

The tf.ones\_like() function creates a tensor filled with ones that has the same shape and data type as an existing tensor.

**Syntax:**

**tf.ones\_like(input, dtype=None, name=None) -> Return Ones Tensor**

**3. tf.ones\_initializer()**

tf.ones\_initializer() is used for initializing variables in layers, especially in neural networks. It initializes all values of a variable to ones.

**Usage in layers:** This is particularly useful when defining weight or bias initializers in neural network layers.

**Syntax:**

**tf.ones\_initializer()**

*tf.ones(): Directly creates a tensor of specified shape filled with ones.*

*tf.ones\_like(): Creates a tensor of ones with the same shape and type as an existing tensor.*

*tf.ones\_initializer(): Used for initializing variables, especially in layers, with ones.*

**Create Zeros vs Zeros \_like vs Zeros \_initializer Tensor :**

**1. tf.zeros**

**Function:** tf.zeros(shape, dtype=tf.float32, name=None)

**Purpose:** Creates a tensor filled with zeros.

**Parameters:**

* **shape:** A list or tuple representing the shape of the tensor to be created.
* **dtype:** The data type of the tensor. Defaults to tf.float32. It can be set to other types like tf.int32, tf.float64, etc.
* **name:** Optional name for the operation.

**Applications:**

* Initializing weights or biases to zero in neural network layers.
* Creating placeholders for tensors during computations.

**2. tf.zeros\_like**

**Function:** tf.zeros\_like(input, dtype=None, name=None, tensor=None)

**Purpose:** Creates a tensor with the same shape and type as a given tensor, but filled with zeros.

**Parameters:**

* **input:** The tensor whose shape and type the new tensor should match.
* **dtype:** Optional. If specified, it overrides the dtype of the new tensor. Otherwise, it defaults to the dtype of input.
* **name:** Optional name for the operation.

**Applications:**

* Creating tensors for operations that need to match the shape and type of another tensor.
* Useful in masking operations or when initializing placeholders.

**3. tf.keras.initializers.Zeros**

**Function:** tf.keras.initializers.Zeros()

**Purpose:** A Keras initializer that creates tensors filled with zeros. It is often used for initializing the weights of neural network layers.

**Parameters:**

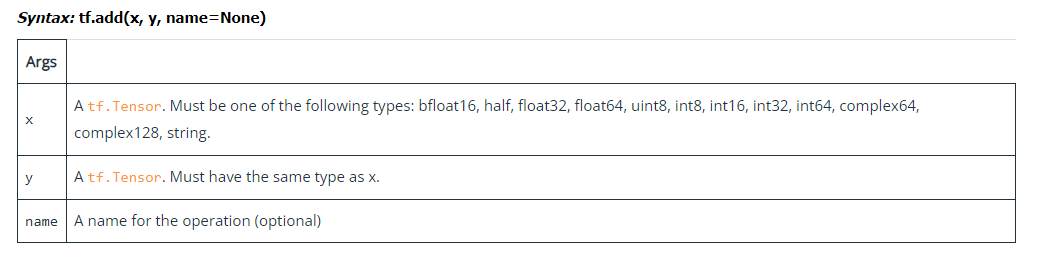
* **seed:** Optional. Seed for random number generator.

**Applications:**

* Used in Keras models for initializing weights of layers, especially when you want a consistent starting point.

**Tensorflow tf.add() Function**

The**tf.add() function**returns the addition of two tf.Tensor objects element wise. The tf.Tensor object represents the multidimensional array of numbers.

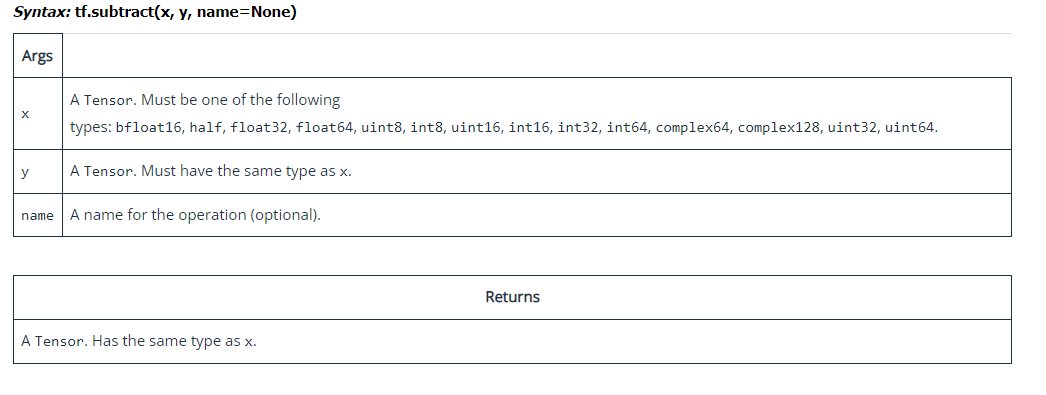


**Tensorflow tf.subtract() Function**

subtract() is used to compute element wise (x-y).

It can subtract list, tuple, scaler, TensorFlow variable/constant/placeholder/SparceMatrix with each other and with scaler and with list/tuple.

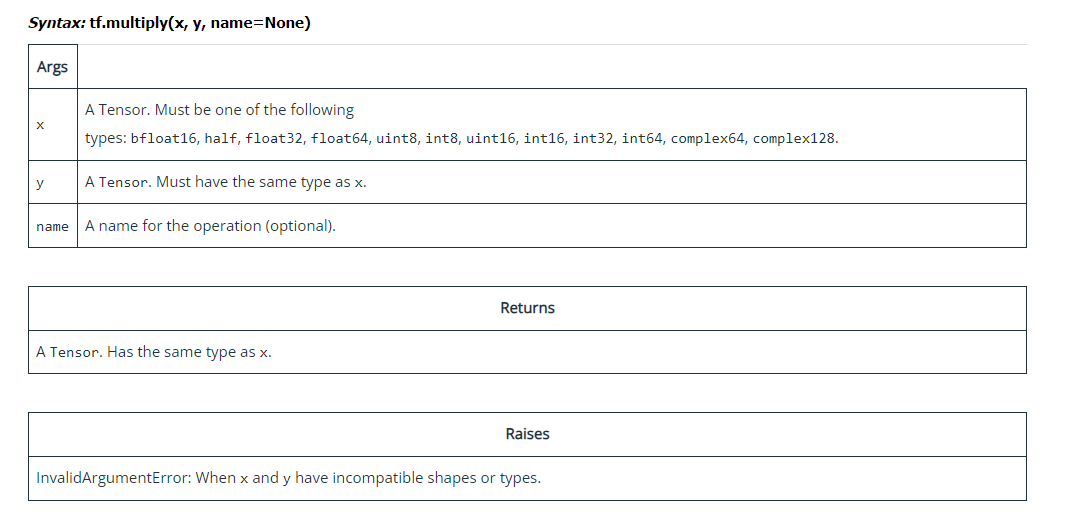
**Note:** – operator can be used to subtract 2 tensors



**Tensorflow tf.multiply() Function**

**multiply()** is used to find element wise x\*y. It supports broadcasting.

It can be multiply list, tuple, scaler, tf variable/constant/placeholder/SparceMatrix with each other and with scaler and with list/tuple.

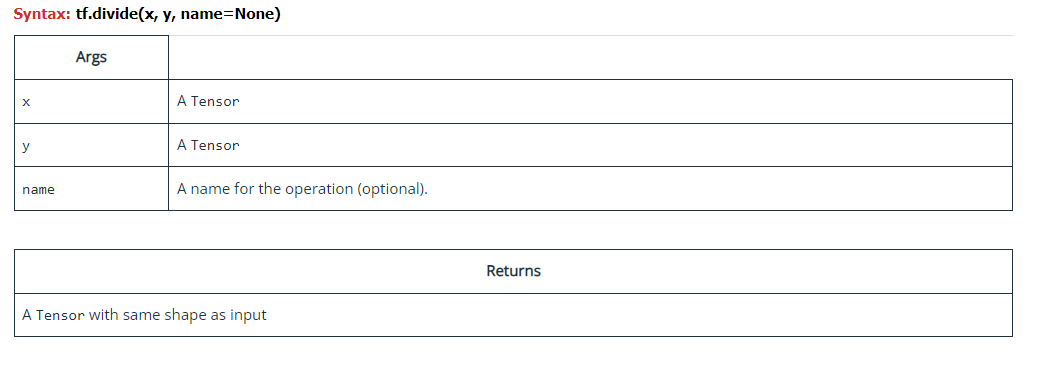


**Tensorflow tf.divide() Function:**

**divide() is used to compute element wise style division of x by y.**

**It can be divide scaler, Numpy array but not with list, tuple.  
tf variable/constant/placeholder/SparceMatrix with each other and with scaler and with list/tuple**

**Note: / or // operator can be use to divide 2 tensors**

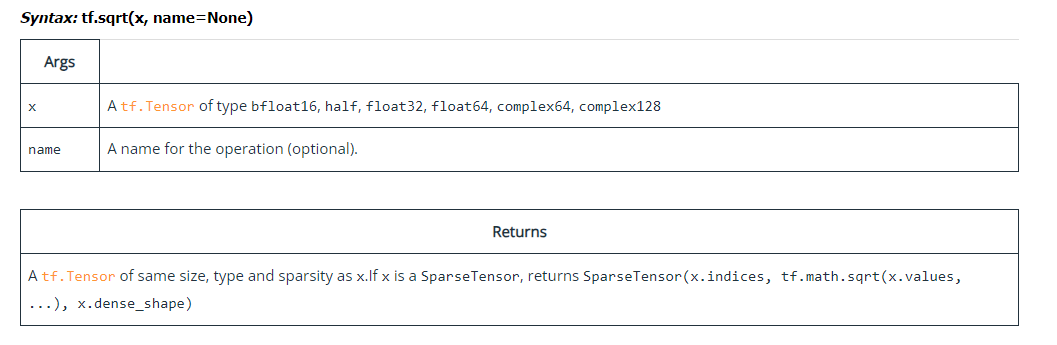


**Tensorflow tf.sqrt() Function:**

**sqrt() is used to compute element wise square root of numpy array and TF tensor,  
It can be give sqrt of list, tuple, scaler.  
It allow tf variable/constant/placeholder/SparceMatrix with Specific datatype**

**Data Type : bfloat16, half, float32, float64, complex64, complex128.**

**Note: Don’t give ‘int’ data type, tf.sqrt() will through error.**



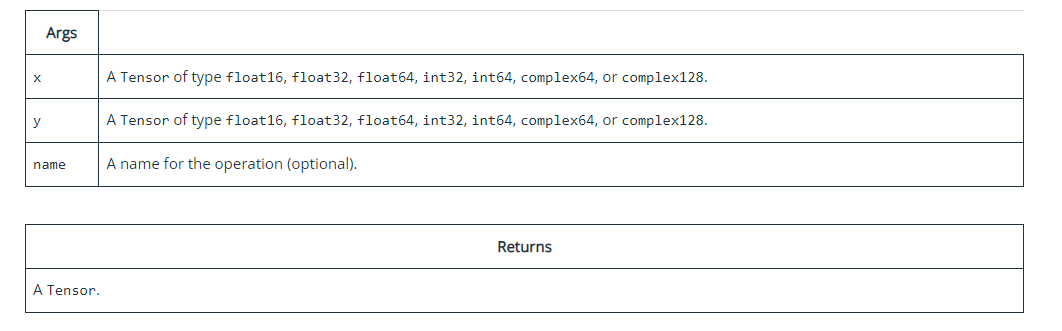
**Tensorflow tf.pow() Function:**

**tf.pow(): Calculate the power of one value to another, Element wise power calculation  
It can be given the power of scaler, Numpy array. It allow tf variable/constant/placeholder/SparceMatrix with Specific datatype**

**Data Type : float16, float32, float64, int32, int64, complex64, or complex128.**

**Note: Don’t pass data in integer format, it will through error**

**Syntax: tf.pow(x, y, name=None)**



**Get Maximum Values:**

**tf.maximum() :** Returns the maximum value by comparing of x and y (i.e. x > y ? x : y) element-wise.  
It work with list, tuple, scaler, Numpy array tf variable/constant/placeholder/SparceMatrix with each other and with scaler and with list/tuple.

**Syntax: tf.maximum(x, y, name=None)**



**Get Minimum Values:**

**tf.minimum() :** Returns the minimum value by comparing of x and y (i.e. x > y ? x : y) element-wise.  
It works with list, tuple, scaler, Numpy array tf variable/constant/placeholder/SparceMatrix with each other and with scaler and with list/tuple.

**Syntax: tf.minimum(x, y, name=None)**

