# Module: tf.math

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math#top_of_page)
* [About Segmentation](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math#about_segmentation)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math#functions)

Math Operations.

**Note:** Functions taking **Tensor** arguments can also take anything accepted by [**tf.convert\_to\_tensor**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/convert_to_tensor).**Note:** Elementwise binary operations in TensorFlow follow [numpy-style broadcasting](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html).

TensorFlow provides a variety of math functions including:

* Basic arithmetic operators and trigonometric functions.
* Special math functions (like: [tf.math.igamma](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/igamma) and [tf.math.zeta](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/zeta))
* Complex number functions (like: [tf.math.imag](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/imag) and [tf.math.angle](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/angle))
* Reductions and scans (like: [tf.math.reduce\_mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean) and [tf.math.cumsum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cumsum))
* Segment functions (like: [tf.math.segment\_sum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_sum))

See: [tf.linalg](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/linalg) for matrix and tensor functions.

## About Segmentation

TensorFlow provides several operations that you can use to perform common math computations on tensor segments. Here a segmentation is a partitioning of a tensor along the first dimension, i.e. it defines a mapping from the first dimension onto segment\_ids. The segment\_ids tensor should be the size of the first dimension, d0, with consecutive IDs in the range 0 to k, where k<d0. In particular, a segmentation of a matrix tensor is a mapping of rows to segments.

#### For example:

c = tf.constant([[1,2,3,4], [-1,-2,-3,-4], [5,6,7,8]])  
tf.math.segment\_sum(c, tf.constant([0, 0, 1]))  
#  ==>  [[0 0 0 0]  
#        [5 6 7 8]]

The standard segment\_\* functions assert that the segment indices are sorted. If you have unsorted indices use the equivalent unsorted\_segment\_ function. Thses functions take an additional argument num\_segments so that the output tensor can be efficiently allocated.

c = tf.constant([[1,2,3,4], [-1,-2,-3,-4], [5,6,7,8]])  
tf.math.unsorted\_segment\_sum(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 6,  8, 10, 12],  
#       [-1, -2, -3, -4]]

## Functions

[abs(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/abs): Computes the absolute value of a tensor.

[accumulate\_n(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/accumulate_n): Returns the element-wise sum of a list of tensors.

[acos(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/acos): Computes acos of x element-wise.

[acosh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/acosh): Computes inverse hyperbolic cosine of x element-wise.

[add(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add): Returns x + y element-wise.

[add\_n(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add_n): Adds all input tensors element-wise.

[angle(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/angle): Returns the element-wise argument of a complex (or real) tensor.

[argmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/argmax): Returns the index with the largest value across axes of a tensor.

[argmin(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/argmin): Returns the index with the smallest value across axes of a tensor.

[asin(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asin): Computes the trignometric inverse sine of x element-wise.

[asinh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asinh): Computes inverse hyperbolic sine of x element-wise.

[atan(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan): Computes the trignometric inverse tangent of x element-wise.

[atan2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan2): Computes arctangent of y/x element-wise, respecting signs of the arguments.

[atanh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atanh): Computes inverse hyperbolic tangent of x element-wise.

[bessel\_i0(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0): Computes the Bessel i0 function of x element-wise.

[bessel\_i0e(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0e): Computes the Bessel i0e function of x element-wise.

[bessel\_i1(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1): Computes the Bessel i1 function of x element-wise.

[bessel\_i1e(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1e): Computes the Bessel i1e function of x element-wise.

[betainc(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/betainc): Compute the regularized incomplete beta integral Ix(a,b).

[bincount(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bincount): Counts the number of occurrences of each value in an integer array.

[ceil(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/ceil): Returns element-wise smallest integer not less than x.

[confusion\_matrix(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/confusion_matrix): Computes the confusion matrix from predictions and labels.

[conj(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/conj): Returns the complex conjugate of a complex number.

[cos(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cos): Computes cos of x element-wise.

[cosh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cosh): Computes hyperbolic cosine of x element-wise.

[count\_nonzero(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/count_nonzero): Computes number of nonzero elements across dimensions of a tensor.

[cumprod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cumprod): Compute the cumulative product of the tensor x along axis.

[cumsum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cumsum): Compute the cumulative sum of the tensor x along axis.

[digamma(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/digamma): Computes Psi, the derivative of Lgamma (the log of the absolute value of

[divide(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/divide): Computes Python style division of x by y.

[divide\_no\_nan(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/divide_no_nan): Computes an unsafe divide which returns 0 if the y is zero.

[equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/equal): Returns the truth value of (x == y) element-wise.

[erf(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/erf): Computes the Gauss error function of x element-wise.

[erfc(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/erfc): Computes the complementary error function of x element-wise.

[exp(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/exp): Computes exponential of x element-wise. y=ex.

[expm1(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/expm1): Computes exponential of x - 1 element-wise.

[floor(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floor): Returns element-wise largest integer not greater than x.

[floordiv(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floordiv): Divides x / y elementwise, rounding toward the most negative integer.

[floormod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floormod): Returns element-wise remainder of division. When x < 0 xor y < 0 is

[greater(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/greater): Returns the truth value of (x > y) element-wise.

[greater\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/greater_equal): Returns the truth value of (x >= y) element-wise.

[igamma(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/igamma): Compute the lower regularized incomplete Gamma function P(a, x).

[igammac(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/igammac): Compute the upper regularized incomplete Gamma function Q(a, x).

[imag(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/imag): Returns the imaginary part of a complex (or real) tensor.

[in\_top\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/in_top_k): Says whether the targets are in the top K predictions.

[invert\_permutation(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/invert_permutation): Computes the inverse permutation of a tensor.

[is\_finite(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_finite): Returns which elements of x are finite.

[is\_inf(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_inf): Returns which elements of x are Inf.

[is\_nan(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_nan): Returns which elements of x are NaN.

[is\_non\_decreasing(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_non_decreasing): Returns True if x is non-decreasing.

[is\_strictly\_increasing(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_strictly_increasing): Returns True if x is strictly increasing.

[l2\_normalize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/l2_normalize): Normalizes along dimension axis using an L2 norm.

[lbeta(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/lbeta): Computes ln(|Beta(x)|), reducing along the last dimension.

[less(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/less): Returns the truth value of (x < y) element-wise.

[less\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/less_equal): Returns the truth value of (x <= y) element-wise.

[lgamma(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/lgamma): Computes the log of the absolute value of Gamma(x) element-wise.

[log(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log): Computes natural logarithm of x element-wise.

[log1p(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log1p): Computes natural logarithm of (1 + x) element-wise.

[log\_sigmoid(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log_sigmoid): Computes log sigmoid of x element-wise.

[log\_softmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/log_softmax): Computes log softmax activations.

[logical\_and(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_and): Returns the truth value of x AND y element-wise.

[logical\_not(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_not): Returns the truth value of NOT x element-wise.

[logical\_or(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_or): Returns the truth value of x OR y element-wise.

[logical\_xor(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_xor): Logical XOR function.

[maximum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/maximum): Returns the max of x and y (i.e. x > y ? x : y) element-wise.

[minimum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/minimum): Returns the min of x and y (i.e. x < y ? x : y) element-wise.

[mod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floormod): Returns element-wise remainder of division. When x < 0 xor y < 0 is

[multiply(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/multiply): Returns x \* y element-wise.

[multiply\_no\_nan(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/multiply_no_nan): Computes the product of x and y and returns 0 if the y is zero, even if x is NaN or infinite.

[negative(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/negative): Computes numerical negative value element-wise.

[nextafter(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/nextafter): Returns the next representable value of x1 in the direction of x2, element-wise.

[not\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/not_equal): Returns the truth value of (x != y) element-wise.

[polygamma(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/polygamma): Compute the polygamma function ψ(n)(x).

[polyval(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/polyval): Computes the elementwise value of a polynomial.

[pow(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/pow): Computes the power of one value to another.

[real(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/real): Returns the real part of a complex (or real) tensor.

[reciprocal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reciprocal): Computes the reciprocal of x element-wise.

[reduce\_all(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/reduce_all): Computes the "logical and" of elements across dimensions of a tensor.

[reduce\_any(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_any): Computes the "logical or" of elements across dimensions of a tensor.

[reduce\_euclidean\_norm(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_euclidean_norm): Computes the Euclidean norm of elements across dimensions of a tensor.

[reduce\_logsumexp(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_logsumexp): Computes log(sum(exp(elements across dimensions of a tensor))).

[reduce\_max(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_max): Computes the maximum of elements across dimensions of a tensor.

[reduce\_mean(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean): Computes the mean of elements across dimensions of a tensor.

[reduce\_min(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_min): Computes the minimum of elements across dimensions of a tensor.

[reduce\_prod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_prod): Computes the product of elements across dimensions of a tensor.

[reduce\_std(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_std): Computes the standard deviation of elements across dimensions of a tensor.

[reduce\_sum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_sum): Computes the sum of elements across dimensions of a tensor.

[reduce\_variance(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_variance): Computes the variance of elements across dimensions of a tensor.

[rint(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/rint): Returns element-wise integer closest to x.

[round(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/round): Rounds the values of a tensor to the nearest integer, element-wise.

[rsqrt(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/rsqrt): Computes reciprocal of square root of x element-wise.

[scalar\_mul(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/scalar_mul): Multiplies a scalar times a Tensor or IndexedSlices object.

[segment\_max(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_max): Computes the maximum along segments of a tensor.

[segment\_mean(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_mean): Computes the mean along segments of a tensor.

[segment\_min(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_min): Computes the minimum along segments of a tensor.

[segment\_prod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_prod): Computes the product along segments of a tensor.

[segment\_sum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_sum): Computes the sum along segments of a tensor.

[sigmoid(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sigmoid): Computes sigmoid of x element-wise.

[sign(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sign): Returns an element-wise indication of the sign of a number.

[sin(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sin): Computes sin of x element-wise.

[sinh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sinh): Computes hyperbolic sine of x element-wise.

[softmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/softmax): Computes softmax activations.

[softplus(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/softplus): Computes softplus: log(exp(features) + 1).

[softsign(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/softsign): Computes softsign: features / (abs(features) + 1).

[sqrt(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sqrt): Computes square root of x element-wise.

[square(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/square): Computes square of x element-wise.

[squared\_difference(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/squared_difference): Returns (x - y)(x - y) element-wise.

[subtract(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/subtract): Returns x - y element-wise.

[tan(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tan): Computes tan of x element-wise.

[tanh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tanh): Computes hyperbolic tangent of x element-wise.

[top\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/top_k): Finds values and indices of the k largest entries for the last dimension.

[truediv(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/truediv): Divides x / y elementwise (using Python 3 division operator semantics).

[unsorted\_segment\_max(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_max): Computes the maximum along segments of a tensor.

[unsorted\_segment\_mean(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_mean): Computes the mean along segments of a tensor.

[unsorted\_segment\_min(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_min): Computes the minimum along segments of a tensor.

[unsorted\_segment\_prod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_prod): Computes the product along segments of a tensor.

[unsorted\_segment\_sqrt\_n(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_sqrt_n): Computes the sum along segments of a tensor divided by the sqrt(N).

[unsorted\_segment\_sum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_sum): Computes the sum along segments of a tensor.

[xdivy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/xdivy): Returns 0 if x == 0, and x / y otherwise, elementwise.

[xlogy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/xlogy): Returns 0 if x == 0, and x \* log(y) otherwise, elementwise.

[zero\_fraction(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/zero_fraction): Returns the fraction of zeros in value.

[zeta(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/zeta): Compute the Hurwitz zeta function ζ(x,q).

# tf.compat.v1.math.log\_softmax

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/math/log_softmax#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/math/log_softmax#aliases)

Computes log softmax activations. (deprecated arguments)

### Aliases:

* tf.compat.v1.math.log\_softmax
* tf.compat.v1.nn.log\_softmax

tf.compat.v1.math.log\_softmax(  
    logits,  
    axis=None,  
    name=None,  
    dim=None  
)

Defined in [python/ops/nn\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/nn_ops.py).

**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(dim)**. They will be removed in a future version. Instructions for updating: dim is deprecated, use axis instead

For each batch i and class j we have

logsoftmax = logits - log(reduce\_sum(exp(logits), axis))

#### Args:

* **logits**: A non-empty Tensor. Must be one of the following types: half, float32, float64.
* **axis**: The dimension softmax would be performed on. The default is -1 which indicates the last dimension.
* **name**: A name for the operation (optional).
* **dim**: Deprecated alias for axis.

#### Returns:

A Tensor. Has the same type as logits. Same shape as logits.

#### Raises:

* **InvalidArgumentError**: if logits is empty or axis is beyond the last dimension of logits.

# tf.compat.v1.math.softmax

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/math/softmax#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/math/softmax#aliases)

Computes softmax activations. (deprecated arguments)

### Aliases:

* tf.compat.v1.math.softmax
* tf.compat.v1.nn.softmax

tf.compat.v1.math.softmax(  
    logits,  
    axis=None,  
    name=None,  
    dim=None  
)

Defined in [python/ops/nn\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/nn_ops.py).

**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(dim)**. They will be removed in a future version. Instructions for updating: dim is deprecated, use axis instead

This function performs the equivalent of

softmax = tf.exp(logits) / tf.reduce\_sum(tf.exp(logits), axis)

#### Args:

* **logits**: A non-empty Tensor. Must be one of the following types: half, float32, float64.
* **axis**: The dimension softmax would be performed on. The default is -1 which indicates the last dimension.
* **name**: A name for the operation (optional).
* **dim**: Deprecated alias for axis.

#### Returns:

A Tensor. Has the same type and shape as logits.

#### Raises:

* **InvalidArgumentError**: if logits is empty or axis is beyond the last dimension of logits.

# tf.math.abs

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/abs#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/abs#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/abs#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/abs#used_in_the_tutorials)

Computes the absolute value of a tensor.

### Aliases:

* tf.RaggedTensor.\_\_abs\_\_
* tf.Tensor.\_\_abs\_\_
* tf.abs
* tf.compat.v1.RaggedTensor.\_\_abs\_\_
* tf.compat.v1.Tensor.\_\_abs\_\_
* tf.compat.v1.abs
* tf.compat.v1.math.abs
* tf.compat.v2.RaggedTensor.\_\_abs\_\_
* tf.compat.v2.Tensor.\_\_abs\_\_
* tf.compat.v2.abs
* tf.compat.v2.math.abs
* tf.math.abs

tf.math.abs(  
    x,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

### Used in the guide:

* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)

### Used in the tutorials:

* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [tf.function](https://www.tensorflow.org/beta/tutorials/eager/tf_function)

Given a tensor of integer or floating-point values, this operation returns a tensor of the same type, where each element contains the absolute value of the corresponding element in the input.

Given a tensor x of complex numbers, this operation returns a tensor of type float32 or float64that is the absolute value of each element in x. All elements in x must be complex numbers of the form a+bj. The absolute value is computed as a2+b2. For example:

x = tf.constant([[-2.25 + 4.75j], [-3.25 + 5.75j]])  
tf.abs(x)  # [5.25594902, 6.60492229]

#### Args:

* **x**: A Tensor or SparseTensor of type float16, float32, float64, int32, int64, complex64 or complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor or SparseTensor the same size, type, and sparsity as x with absolute values. Note, for complex64 or complex128 input, the returned Tensor will be of type float32 or float64, respectively.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.abs(x.values, ...), x.dense\_shape)

# tf.math.accumulate\_n

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/accumulate_n#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/accumulate_n#aliases)

Returns the element-wise sum of a list of tensors.

### Aliases:

* tf.compat.v1.accumulate\_n
* tf.compat.v1.math.accumulate\_n
* tf.compat.v2.math.accumulate\_n
* tf.math.accumulate\_n

tf.math.accumulate\_n(  
    inputs,  
    shape=None,  
    tensor\_dtype=None,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Optionally, pass shape and tensor\_dtype for shape and type checking, otherwise, these are inferred.

accumulate\_n performs the same operation as [tf.math.add\_n](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add_n), but does not wait for all of its inputs to be ready before beginning to sum. This approach can save memory if inputs are ready at different times, since minimum temporary storage is proportional to the output size rather than the inputs' size.

accumulate\_n is differentiable (but wasn't previous to TensorFlow 1.7).

#### For example:

a = tf.constant([[1, 2], [3, 4]])  
b = tf.constant([[5, 0], [0, 6]])  
tf.math.accumulate\_n([a, b, a])  # [[7, 4], [6, 14]]  
  
# Explicitly pass shape and type  
tf.math.accumulate\_n([a, b, a], shape=[2, 2], tensor\_dtype=tf.int32)  
                                                               # [[7,  4],  
                                                               #  [6, 14]]

#### Args:

* **inputs**: A list of Tensor objects, each with same shape and type.
* **shape**: Expected shape of elements of inputs (optional). Also controls the output shape of this op, which may affect type inference in other ops. A value of None means "infer the input shape from the shapes in inputs".
* **tensor\_dtype**: Expected data type of inputs (optional). A value of None means "infer the input dtype from inputs[0]".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of same shape and type as the elements of inputs.

#### Raises:

* **ValueError**: If inputs don't all have same shape and dtype or the shape cannot be inferred.

# tf.math.acos

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/acos#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/acos#aliases)

Computes acos of x element-wise.

### Aliases:

* tf.acos
* tf.compat.v1.acos
* tf.compat.v1.math.acos
* tf.compat.v2.acos
* tf.compat.v2.math.acos
* tf.math.acos

tf.math.acos(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.acosh

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/acosh#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/acosh#aliases)

Computes inverse hyperbolic cosine of x element-wise.

### Aliases:

* tf.acosh
* tf.compat.v1.acosh
* tf.compat.v1.math.acosh
* tf.compat.v2.acosh
* tf.compat.v2.math.acosh
* tf.math.acosh

tf.math.acosh(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.add

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add#used_in_the_tutorials)

Returns x + y element-wise.

### Aliases:

* tf.RaggedTensor.\_\_add\_\_
* tf.add
* tf.compat.v1.RaggedTensor.\_\_add\_\_
* tf.compat.v1.add
* tf.compat.v1.math.add
* tf.compat.v2.RaggedTensor.\_\_add\_\_
* tf.compat.v2.add
* tf.compat.v2.math.add
* tf.math.add

tf.math.add(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)

### Used in the tutorials:

* [Tensors and Operations](https://www.tensorflow.org/beta/tutorials/eager/basics)

NOTE: math.add supports broadcasting. AddN does not. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, uint8, int8, int16, int32, int64, complex64, complex128, string.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.add\_n

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add_n#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add_n#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add_n#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add_n#used_in_the_tutorials)

Adds all input tensors element-wise.

### Aliases:

* tf.add\_n
* tf.compat.v1.add\_n
* tf.compat.v1.math.add\_n
* tf.compat.v2.add\_n
* tf.compat.v2.math.add\_n
* tf.math.add\_n

tf.math.add\_n(  
    inputs,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [Using GPUs](https://www.tensorflow.org/beta/guide/using_gpu)

### Used in the tutorials:

* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)

Converts IndexedSlices objects into dense tensors prior to adding.

[tf.math.add\_n](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add_n) performs the same operation as [tf.math.accumulate\_n](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/accumulate_n), but it waits for all of its inputs to be ready before beginning to sum. This buffering can result in higher memory consumption when inputs are ready at different times, since the minimum temporary storage required is proportional to the input size rather than the output size.

This op does not [broadcast](https://docs.scipy.org/doc/numpy-1.13.0/user/basics.broadcasting.html) its inputs. If you need broadcasting, use [tf.math.add](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add) (or the + operator) instead.

#### For example:

a = tf.constant([[3, 5], [4, 8]])  
b = tf.constant([[1, 6], [2, 9]])  
tf.math.add\_n([a, b, a])  # [[7, 16], [10, 25]]

#### Args:

* **inputs**: A list of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) or [tf.IndexedSlices](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/IndexedSlices) objects, each with same shape and type.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of same shape and type as the elements of inputs.

#### Raises:

* **ValueError**: If inputs don't all have same shape and dtype or the shape cannot be inferred.

# tf.math.angle

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/angle#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/angle#aliases)

Returns the element-wise argument of a complex (or real) tensor.

### Aliases:

* tf.compat.v1.angle
* tf.compat.v1.math.angle
* tf.compat.v2.math.angle
* tf.math.angle

tf.math.angle(  
    input,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Given a tensor input, this operation returns a tensor of type float that is the argument of each element in input considered as a complex number.

The elements in input are considered to be complex numbers of the form a+bj, where a is the real part and b is the imaginary part. If input is real then b is zero by definition.

The argument returned by this function is of the form atan2(b,a). If input is real, a tensor of all zeros is returned.

#### For example:

input = tf.constant([-2.25 + 4.75j, 3.25 + 5.75j], dtype=tf.complex64)  
tf.math.angle(input).numpy()  
# ==> array([2.0131705, 1.056345 ], dtype=float32)

#### Args:

* **input**: A Tensor. Must be one of the following types: float, double, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type float32 or float64.

# tf.math.argmax

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/argmax#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/argmax#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/argmax#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/argmax#used_in_the_tutorials)

Returns the index with the largest value across axes of a tensor.

### Aliases:

* tf.argmax
* tf.compat.v2.argmax
* tf.compat.v2.math.argmax
* tf.math.argmax

tf.math.argmax(  
    input,  
    axis=None,  
    output\_type=tf.dtypes.int64,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

### Used in the tutorials:

* [Custom training: walkthrough](https://www.tensorflow.org/beta/tutorials/eager/custom_training_walkthrough)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

Note that in case of ties the identity of the return value is not guaranteed.

#### Args:

* **input**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **axis**: A Tensor. Must be one of the following types: int32, int64. int32 or int64, must be in the range -rank(input), rank(input)). Describes which axis of the input Tensor to reduce across. For vectors, use axis = 0.
* **output\_type**: An optional [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) from: tf.int32, tf.int64. Defaults to [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type output\_type.

#### Usage:

import tensorflow as tf  
a = [1, 10, 26.9, 2.8, 166.32, 62.3]  
b = tf.math.argmax(input = a)  
c = tf.keras.backend.eval(b)  
# c = 4  
# here a[4] = 166.32 which is the largest element of a across axis 0

# tf.math.argmin

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/argmin#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/argmin#aliases)

Returns the index with the smallest value across axes of a tensor.

### Aliases:

* tf.argmin
* tf.compat.v2.argmin
* tf.compat.v2.math.argmin
* tf.math.argmin

tf.math.argmin(  
    input,  
    axis=None,  
    output\_type=tf.dtypes.int64,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Note that in case of ties the identity of the return value is not guaranteed.

#### Args:

* **input**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **axis**: A Tensor. Must be one of the following types: int32, int64. int32 or int64, must be in the range -rank(input), rank(input)). Describes which axis of the input Tensor to reduce across. For vectors, use axis = 0.
* **output\_type**: An optional [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) from: tf.int32, tf.int64. Defaults to [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type output\_type.

#### Usage:

import tensorflow as tf  
a = [1, 10, 26.9, 2.8, 166.32, 62.3]  
b = tf.math.argmin(input = a)  
c = tf.keras.backend.eval(b)  
# c = 0  
# here a[0] = 1 which is the smallest element of a across axis 0

# tf.math.asin

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asin#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asin#aliases)

Computes the trignometric inverse sine of x element-wise.

### Aliases:

* tf.asin
* tf.compat.v1.asin
* tf.compat.v1.math.asin
* tf.compat.v2.asin
* tf.compat.v2.math.asin
* tf.math.asin

tf.math.asin(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

The [tf.math.asin](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asin) operation returns the inverse of [tf.math.sin](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sin), such that if y = tf.math.sin(x) then, x = tf.math.asin(y).

**Note**: The output of [tf.math.asin](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asin) will lie within the invertible range of sine, i.e [-pi/2, pi/2].

#### For example:

# Note: [1.047, 0.785] ~= [(pi/3), (pi/4)]  
x = tf.constant([1.047, 0.785])  
y = tf.math.sin(x) # [0.8659266, 0.7068252]  
  
tf.math.asin(y) # [1.047, 0.785] = x

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.asinh

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asinh#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asinh#aliases)

Computes inverse hyperbolic sine of x element-wise.

### Aliases:

* tf.asinh
* tf.compat.v1.asinh
* tf.compat.v1.math.asinh
* tf.compat.v2.asinh
* tf.compat.v2.math.asinh
* tf.math.asinh

tf.math.asinh(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.atan

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan#aliases)

Computes the trignometric inverse tangent of x element-wise.

### Aliases:

* tf.atan
* tf.compat.v1.atan
* tf.compat.v1.math.atan
* tf.compat.v2.atan
* tf.compat.v2.math.atan
* tf.math.atan

tf.math.atan(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

The [tf.math.atan](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan) operation returns the inverse of [tf.math.tan](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tan), such that if y = tf.math.tan(x) then, x = tf.math.atan(y).

**Note**: The output of [tf.math.atan](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan) will lie within the invertible range of tan, i.e (-pi/2, pi/2).

#### For example:

# Note: [1.047, 0.785] ~= [(pi/3), (pi/4)]  
x = tf.constant([1.047, 0.785])  
y = tf.math.tan(x) # [1.731261, 0.99920404]  
  
tf.math.atan(y) # [1.047, 0.785] = x

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.atan2

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan2#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan2#aliases)

Computes arctangent of y/x element-wise, respecting signs of the arguments.

### Aliases:

* tf.atan2
* tf.compat.v1.atan2
* tf.compat.v1.math.atan2
* tf.compat.v2.atan2
* tf.compat.v2.math.atan2
* tf.math.atan2

tf.math.atan2(  
    y,  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

This is the angle ( \theta \in [-\pi, \pi] ) such that [ x = r \cos(\theta) ] and [ y = r \sin(\theta) ] where (r = \sqrt(x^2 + y^2) ).

#### Args:

* **y**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **x**: A Tensor. Must have the same type as y.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as y.

# tf.math.atanh

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atanh#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atanh#aliases)

Computes inverse hyperbolic tangent of x element-wise.

### Aliases:

* tf.atanh
* tf.compat.v1.atanh
* tf.compat.v1.math.atanh
* tf.compat.v2.atanh
* tf.compat.v2.math.atanh
* tf.math.atanh

tf.math.atanh(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.bessel\_i0

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0#aliases)

Computes the Bessel i0 function of x element-wise.

### Aliases:

* tf.compat.v1.math.bessel\_i0
* tf.compat.v2.math.bessel\_i0
* tf.math.bessel\_i0

tf.math.bessel\_i0(  
    x,  
    name=None  
)

Defined in [python/ops/special\_math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/special_math_ops.py).

Modified Bessel function of order 0.

It is preferable to use the numerically stabler function i0e(x) instead.

#### Args:

* **x**: A Tensor or SparseTensor. Must be one of the following types: half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor or SparseTensor, respectively. Has the same type as x.

#### Scipy Compatibility

Equivalent to scipy.special.i0

# tf.math.bessel\_i0e

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0e#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0e#aliases)

Computes the Bessel i0e function of x element-wise.

### Aliases:

* tf.compat.v1.math.bessel\_i0e
* tf.compat.v2.math.bessel\_i0e
* tf.math.bessel\_i0e

tf.math.bessel\_i0e(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

Exponentially scaled modified Bessel function of order 0 defined as bessel\_i0e(x) = exp(-abs(x)) bessel\_i0(x).

This function is faster and numerically stabler than bessel\_i0(x).

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.bessel\_i0e(x.values, ...), x.dense\_shape)

# tf.math.bessel\_i1

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1#aliases)

Computes the Bessel i1 function of x element-wise.

### Aliases:

* tf.compat.v1.math.bessel\_i1
* tf.compat.v2.math.bessel\_i1
* tf.math.bessel\_i1

tf.math.bessel\_i1(  
    x,  
    name=None  
)

Defined in [python/ops/special\_math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/special_math_ops.py).

Modified Bessel function of order 1.

It is preferable to use the numerically stabler function i1e(x) instead.

#### Args:

* **x**: A Tensor or SparseTensor. Must be one of the following types: half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor or SparseTensor, respectively. Has the same type as x.

#### Scipy Compatibility

Equivalent to scipy.special.i1

# tf.math.bessel\_i1e

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1e#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1e#aliases)

Computes the Bessel i1e function of x element-wise.

### Aliases:

* tf.compat.v1.math.bessel\_i1e
* tf.compat.v2.math.bessel\_i1e
* tf.math.bessel\_i1e

tf.math.bessel\_i1e(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

Exponentially scaled modified Bessel function of order 0 defined as bessel\_i1e(x) = exp(-abs(x)) bessel\_i1(x).

This function is faster and numerically stabler than bessel\_i1(x).

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.bessel\_i1e(x.values, ...), x.dense\_shape)

# tf.math.betainc

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/betainc#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/betainc#aliases)

Compute the regularized incomplete beta integral Ix(a,b).

### Aliases:

* tf.compat.v1.betainc
* tf.compat.v1.math.betainc
* tf.compat.v2.math.betainc
* tf.math.betainc

tf.math.betainc(  
    a,  
    b,  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

The regularized incomplete beta integral is defined as:

Ix(a,b)=B(x;a,b)B(a,b)

where

B(x;a,b)=∫0xta−1(1−t)b−1dt

is the incomplete beta function and B(a,b) is the complete beta function.

#### Args:

* **a**: A Tensor. Must be one of the following types: float32, float64.
* **b**: A Tensor. Must have the same type as a.
* **x**: A Tensor. Must have the same type as a.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as a.

# tf.math.bincount

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bincount#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bincount#aliases)

Counts the number of occurrences of each value in an integer array.

### Aliases:

* tf.compat.v2.math.bincount
* tf.math.bincount

tf.math.bincount(  
    arr,  
    weights=None,  
    minlength=None,  
    maxlength=None,  
    dtype=tf.dtypes.int32,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

If minlength and maxlength are not given, returns a vector with length tf.reduce\_max(arr) + 1if arr is non-empty, and length 0 otherwise. If weights are non-None, then index i of the output stores the sum of the value in weights at each index where the corresponding value in arr is i.

#### Args:

* **arr**: An int32 tensor of non-negative values.
* **weights**: If non-None, must be the same shape as arr. For each value in arr, the bin will be incremented by the corresponding weight instead of 1.
* **minlength**: If given, ensures the output has length at least minlength, padding with zeros at the end if necessary.
* **maxlength**: If given, skips values in arr that are equal or greater than maxlength, ensuring that the output has length at most maxlength.
* **dtype**: If weights is None, determines the type of the output bins.
* **name**: A name scope for the associated operations (optional).

#### Returns:

A vector with the same dtype as weights or the given dtype. The bin values.

# tf.math.ceil

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/ceil#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/ceil#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/ceil#used_in_the_tutorials)

Returns element-wise smallest integer not less than x.

### Aliases:

* tf.compat.v1.ceil
* tf.compat.v1.math.ceil
* tf.compat.v2.math.ceil
* tf.math.ceil

tf.math.ceil(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the tutorials:

* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.confusion\_matrix

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/confusion_matrix#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/confusion_matrix#aliases)

Computes the confusion matrix from predictions and labels.

### Aliases:

* tf.compat.v2.math.confusion\_matrix
* tf.math.confusion\_matrix

tf.math.confusion\_matrix(  
    labels,  
    predictions,  
    num\_classes=None,  
    weights=None,  
    dtype=tf.dtypes.int32,  
    name=None  
)

Defined in [python/ops/confusion\_matrix.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/confusion_matrix.py).

The matrix columns represent the prediction labels and the rows represent the real labels. The confusion matrix is always a 2-D array of shape [n, n], where n is the number of valid labels for a given classification task. Both prediction and labels must be 1-D arrays of the same shape in order for this function to work.

If num\_classes is None, then num\_classes will be set to one plus the maximum value in either predictions or labels. Class labels are expected to start at 0. For example, if num\_classes is 3, then the possible labels would be [0, 1, 2].

If weights is not None, then each prediction contributes its corresponding weight to the total value of the confusion matrix cell.

#### For example:

  tf.math.confusion\_matrix([1, 2, 4], [2, 2, 4]) ==>  
      [[0 0 0 0 0]  
       [0 0 1 0 0]  
       [0 0 1 0 0]  
       [0 0 0 0 0]  
       [0 0 0 0 1]]

Note that the possible labels are assumed to be [0, 1, 2, 3, 4], resulting in a 5x5 confusion matrix.

#### Args:

* **labels**: 1-D Tensor of real labels for the classification task.
* **predictions**: 1-D Tensor of predictions for a given classification.
* **num\_classes**: The possible number of labels the classification task can have. If this value is not provided, it will be calculated using both predictions and labels array.
* **weights**: An optional Tensor whose shape matches predictions.
* **dtype**: Data type of the confusion matrix.
* **name**: Scope name.

#### Returns:

A Tensor of type dtype with shape [n, n] representing the confusion matrix, where n is the number of possible labels in the classification task.

#### Raises:

* **ValueError**: If both predictions and labels are not 1-D vectors and have mismatched shapes, or if weights is not None and its shape doesn't match predictions.

# tf.math.conj

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/conj#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/conj#aliases)

Returns the complex conjugate of a complex number.

### Aliases:

* tf.compat.v1.conj
* tf.compat.v1.math.conj
* tf.compat.v2.math.conj
* tf.math.conj

tf.math.conj(  
    x,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Given a tensor input of complex numbers, this operation returns a tensor of complex numbers that are the complex conjugate of each element in input. The complex numbers in input must be of the form a+bj, where a is the real part and b is the imaginary part.

The complex conjugate returned by this operation is of the form a−bj.

#### For example:

# tensor 'input' is [-2.25 + 4.75j, 3.25 + 5.75j]

tf.math.conj(input) ==> [-2.25 - 4.75j, 3.25 - 5.75j]

If x is real, it is returned unchanged.

#### Args:

* **x**: Tensor to conjugate. Must have numeric or variant type.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor that is the conjugate of x (with the same type).

#### Raises:

* **TypeError**: If x is not a numeric tensor.

# tf.math.cos

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cos#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cos#aliases)

Computes cos of x element-wise.

### Aliases:

* tf.compat.v1.cos
* tf.compat.v1.math.cos
* tf.compat.v2.cos
* tf.compat.v2.math.cos
* tf.cos
* tf.math.cos

tf.math.cos(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.cosh

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cosh#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cosh#aliases)

Computes hyperbolic cosine of x element-wise.

### Aliases:

* tf.compat.v1.cosh
* tf.compat.v1.math.cosh
* tf.compat.v2.cosh
* tf.compat.v2.math.cosh
* tf.cosh
* tf.math.cosh

tf.math.cosh(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.count\_nonzero

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/count_nonzero#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/count_nonzero#aliases)

Computes number of nonzero elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.count\_nonzero
* tf.math.count\_nonzero

tf.math.count\_nonzero(  
    input,  
    axis=None,  
    keepdims=None,  
    dtype=tf.dtypes.int64,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Reduces input along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis has no entries, all dimensions are reduced, and a tensor with a single element is returned.

**NOTE** Floating point comparison to zero is done by exact floating point equality check. Small values are **not** rounded to zero for purposes of the nonzero check.

#### For example:

x = tf.constant([[0, 1, 0], [1, 1, 0]])  
tf.math.count\_nonzero(x)  # 3  
tf.math.count\_nonzero(x, 0)  # [1, 2, 0]  
tf.math.count\_nonzero(x, 1)  # [1, 2]  
tf.math.count\_nonzero(x, 1, keepdims=True)  # [[1], [2]]  
tf.math.count\_nonzero(x, [0, 1])  # 3

**NOTE** Strings are compared against zero-length empty string "". Any string with a size greater than zero is already considered as nonzero.

#### For example:

x = tf.constant(["", "a", "  ", "b", ""])  
tf.math.count\_nonzero(x) # 3, with "a", "  ", and "b" as nonzero strings.

#### Args:

* **input**: The tensor to reduce. Should be of numeric type, bool, or string.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input), rank(input)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **dtype**: The output dtype; defaults to [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64).
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor (number of nonzero values).

# tf.math.cumprod

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cumprod#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cumprod#aliases)

Compute the cumulative product of the tensor x along axis.

### Aliases:

* tf.compat.v1.cumprod
* tf.compat.v1.math.cumprod
* tf.compat.v2.math.cumprod
* tf.math.cumprod

tf.math.cumprod(  
    x,  
    axis=0,  
    exclusive=False,  
    reverse=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

By default, this op performs an inclusive cumprod, which means that the first element of the input is identical to the first element of the output:

tf.math.cumprod([a, b, c])  # [a, a \* b, a \* b \* c]

By setting the exclusive kwarg to True, an exclusive cumprod is performed instead:

tf.math.cumprod([a, b, c], exclusive=True)  # [1, a, a \* b]

By setting the reverse kwarg to True, the cumprod is performed in the opposite direction:

tf.math.cumprod([a, b, c], reverse=True)  # [a \* b \* c, b \* c, c]

This is more efficient than using separate [tf.reverse](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/reverse) ops. The reverse and exclusive kwargs can also be combined:

tf.math.cumprod([a, b, c], exclusive=True, reverse=True)  # [b \* c, c, 1]

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int64, int32, uint8, uint16, int16, int8, complex64, complex128, qint8, quint8, qint32, half.
* **axis**: A Tensor of type int32 (default: 0). Must be in the range [-rank(x), rank(x)).
* **exclusive**: If True, perform exclusive cumprod.
* **reverse**: A bool (default: False).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.cumsum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cumsum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cumsum#aliases)

Compute the cumulative sum of the tensor x along axis.

### Aliases:

* tf.compat.v1.cumsum
* tf.compat.v1.math.cumsum
* tf.compat.v2.cumsum
* tf.compat.v2.math.cumsum
* tf.cumsum
* tf.math.cumsum

tf.math.cumsum(  
    x,  
    axis=0,  
    exclusive=False,  
    reverse=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

By default, this op performs an inclusive cumsum, which means that the first element of the input is identical to the first element of the output:

tf.cumsum([a, b, c])  # [a, a + b, a + b + c]

By setting the exclusive kwarg to True, an exclusive cumsum is performed instead:

tf.cumsum([a, b, c], exclusive=True)  # [0, a, a + b]

By setting the reverse kwarg to True, the cumsum is performed in the opposite direction:

tf.cumsum([a, b, c], reverse=True)  # [a + b + c, b + c, c]

This is more efficient than using separate [tf.reverse](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/reverse) ops.

The reverse and exclusive kwargs can also be combined:

tf.cumsum([a, b, c], exclusive=True, reverse=True)  # [b + c, c, 0]

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int64, int32, uint8, uint16, int16, int8, complex64, complex128, qint8, quint8, qint32, half.
* **axis**: A Tensor of type int32 (default: 0). Must be in the range [-rank(x), rank(x)).
* **exclusive**: If True, perform exclusive cumsum.
* **reverse**: A bool (default: False).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.digamma

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/digamma#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/digamma#aliases)

Computes Psi, the derivative of Lgamma (the log of the absolute value of

### Aliases:

* tf.compat.v1.digamma
* tf.compat.v1.math.digamma
* tf.compat.v2.math.digamma
* tf.math.digamma

tf.math.digamma(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

Gamma(x)), element-wise.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.divide

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/divide#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/divide#aliases)

Computes Python style division of x by y.

### Aliases:

* tf.compat.v1.divide
* tf.compat.v1.math.divide
* tf.compat.v2.divide
* tf.compat.v2.math.divide
* tf.divide
* tf.math.divide

tf.math.divide(  
    x,  
    y,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

# tf.math.divide\_no\_nan

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/divide_no_nan#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/divide_no_nan#aliases)

Computes an unsafe divide which returns 0 if the y is zero.

### Aliases:

* tf.compat.v1.div\_no\_nan
* tf.compat.v1.math.divide\_no\_nan
* tf.compat.v2.math.divide\_no\_nan
* tf.math.divide\_no\_nan

tf.math.divide\_no\_nan(  
    x,  
    y,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64.
* **y**: A Tensor whose dtype is compatible with x.
* **name**: A name for the operation (optional).

#### Returns:

The element-wise value of the x divided by y.

# tf.math.equal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/equal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/equal#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/equal#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/equal#used_in_the_tutorials)

Returns the truth value of (x == y) element-wise.

### Aliases:

* tf.compat.v1.equal
* tf.compat.v1.math.equal
* tf.compat.v2.equal
* tf.compat.v2.math.equal
* tf.equal
* tf.math.equal

tf.math.equal(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the guide:

* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load CSV with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/csv)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)
* [tf.function](https://www.tensorflow.org/beta/tutorials/eager/tf_function)

NOTE: math.equal supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, uint8, int8, int16, int32, int64, complex64, quint8, qint8, qint32, string, bool, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.erf

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/erf#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/erf#aliases)

Computes the Gauss error function of x element-wise.

### Aliases:

* tf.compat.v1.erf
* tf.compat.v1.math.erf
* tf.compat.v2.math.erf
* tf.math.erf

tf.math.erf(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.erf(x.values, ...), x.dense\_shape)

# tf.math.erfc

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/erfc#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/erfc#aliases)

Computes the complementary error function of x element-wise.

### Aliases:

* tf.compat.v1.erfc
* tf.compat.v1.math.erfc
* tf.compat.v2.math.erfc
* tf.math.erfc

tf.math.erfc(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.exp

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/exp#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/exp#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/exp#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/exp#used_in_the_tutorials)

Computes exponential of x element-wise. y=ex.

### Aliases:

* tf.compat.v1.exp
* tf.compat.v1.math.exp
* tf.compat.v2.exp
* tf.compat.v2.math.exp
* tf.exp
* tf.math.exp

tf.math.exp(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.expm1

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/expm1#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/expm1#aliases)

Computes exponential of x - 1 element-wise.

### Aliases:

* tf.compat.v1.expm1
* tf.compat.v1.math.expm1
* tf.compat.v2.math.expm1
* tf.math.expm1

tf.math.expm1(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

I.e., y=(exp⁡x)−1.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.floor

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floor#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floor#aliases)

Returns element-wise largest integer not greater than x.

### Aliases:

* tf.compat.v1.floor
* tf.compat.v1.math.floor
* tf.compat.v2.floor
* tf.compat.v2.math.floor
* tf.floor
* tf.math.floor

tf.math.floor(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.floordiv

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floordiv#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floordiv#aliases)

Divides x / y elementwise, rounding toward the most negative integer.

### Aliases:

* tf.RaggedTensor.\_\_floordiv\_\_
* tf.compat.v1.RaggedTensor.\_\_floordiv\_\_
* tf.compat.v1.floordiv
* tf.compat.v1.math.floordiv
* tf.compat.v2.RaggedTensor.\_\_floordiv\_\_
* tf.compat.v2.math.floordiv
* tf.math.floordiv

tf.math.floordiv(  
    x,  
    y,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

The same as tf.compat.v1.div(x,y) for integers, but uses tf.floor(tf.compat.v1.div(x,y))for floating point arguments so that the result is always an integer (though possibly an integer represented as floating point). This op is generated by x // y floor division in Python 3 and in Python 2.7 with from \_\_future\_\_ import division.

x and y must have the same type, and the result will have the same type as well.

#### Args:

* **x**: Tensor numerator of real numeric type.
* **y**: Tensor denominator of real numeric type.
* **name**: A name for the operation (optional).

#### Returns:

x / y rounded down.

#### Raises:

* **TypeError**: If the inputs are complex.

# tf.math.floormod

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floormod#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floormod#aliases)

Returns element-wise remainder of division. When x < 0 xor y < 0 is

### Aliases:

* tf.RaggedTensor.\_\_mod\_\_
* tf.compat.v1.RaggedTensor.\_\_mod\_\_
* tf.compat.v1.floormod
* tf.compat.v1.math.floormod
* tf.compat.v1.math.mod
* tf.compat.v1.mod
* tf.compat.v2.RaggedTensor.\_\_mod\_\_
* tf.compat.v2.math.floormod
* tf.compat.v2.math.mod
* tf.math.floormod
* tf.math.mod

tf.math.floormod(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

true, this follows Python semantics in that the result here is consistent with a flooring divide. E.g. floor(x / y) \* y + mod(x, y) = x.

NOTE: math.floormod supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: int32, int64, bfloat16, half, float32, float64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.greater

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/greater#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/greater#aliases)

Returns the truth value of (x > y) element-wise.

### Aliases:

* tf.RaggedTensor.\_\_gt\_\_
* tf.Tensor.\_\_gt\_\_
* tf.compat.v1.RaggedTensor.\_\_gt\_\_
* tf.compat.v1.Tensor.\_\_gt\_\_
* tf.compat.v1.greater
* tf.compat.v1.math.greater
* tf.compat.v2.RaggedTensor.\_\_gt\_\_
* tf.compat.v2.Tensor.\_\_gt\_\_
* tf.compat.v2.greater
* tf.compat.v2.math.greater
* tf.greater
* tf.math.greater

tf.math.greater(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

NOTE: math.greater supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.greater\_equal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/greater_equal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/greater_equal#aliases)

Returns the truth value of (x >= y) element-wise.

### Aliases:

* tf.RaggedTensor.\_\_ge\_\_
* tf.Tensor.\_\_ge\_\_
* tf.compat.v1.RaggedTensor.\_\_ge\_\_
* tf.compat.v1.Tensor.\_\_ge\_\_
* tf.compat.v1.greater\_equal
* tf.compat.v1.math.greater\_equal
* tf.compat.v2.RaggedTensor.\_\_ge\_\_
* tf.compat.v2.Tensor.\_\_ge\_\_
* tf.compat.v2.greater\_equal
* tf.compat.v2.math.greater\_equal
* tf.greater\_equal
* tf.math.greater\_equal

tf.math.greater\_equal(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

NOTE: math.greater\_equal supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.igamma

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/igamma#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/igamma#aliases)

Compute the lower regularized incomplete Gamma function P(a, x).

### Aliases:

* tf.compat.v1.igamma
* tf.compat.v1.math.igamma
* tf.compat.v2.math.igamma
* tf.math.igamma

tf.math.igamma(  
    a,  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

The lower regularized incomplete Gamma function is defined as:

P(a,x)=gamma(a,x)/Gamma(a)=1−Q(a,x)

where

gamma(a,x)=int0xta−1exp(−t)dt

is the lower incomplete Gamma function.

Note, above Q(a, x) (Igammac) is the upper regularized complete Gamma function.

#### Args:

* **a**: A Tensor. Must be one of the following types: float32, float64.
* **x**: A Tensor. Must have the same type as a.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as a.

# tf.math.igammac

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/igammac#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/igammac#aliases)

Compute the upper regularized incomplete Gamma function Q(a, x).

### Aliases:

* tf.compat.v1.igammac
* tf.compat.v1.math.igammac
* tf.compat.v2.math.igammac
* tf.math.igammac

tf.math.igammac(  
    a,  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

The upper regularized incomplete Gamma function is defined as:

Q(a,x)=Gamma(a,x)/Gamma(a)=1−P(a,x)

where

Gamma(a,x)=intx∞ta−1exp(−t)dt

is the upper incomplete Gama function.

Note, above P(a, x) (Igamma) is the lower regularized complete Gamma function.

#### Args:

* **a**: A Tensor. Must be one of the following types: float32, float64.
* **x**: A Tensor. Must have the same type as a.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as a.

# tf.math.imag

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/imag#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/imag#aliases)

Returns the imaginary part of a complex (or real) tensor.

### Aliases:

* tf.compat.v1.imag
* tf.compat.v1.math.imag
* tf.compat.v2.math.imag
* tf.math.imag

tf.math.imag(  
    input,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Given a tensor input, this operation returns a tensor of type float that is the imaginary part of each element in input considered as a complex number. If input is real, a tensor of all zeros is returned.

#### For example:

x = tf.constant([-2.25 + 4.75j, 3.25 + 5.75j])  
tf.math.imag(x)  # [4.75, 5.75]

#### Args:

* **input**: A Tensor. Must be one of the following types: float, double, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type float32 or float64.

# tf.math.invert\_permutation

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/invert_permutation#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/invert_permutation#aliases)

Computes the inverse permutation of a tensor.

### Aliases:

* tf.compat.v1.invert\_permutation
* tf.compat.v1.math.invert\_permutation
* tf.compat.v2.math.invert\_permutation
* tf.math.invert\_permutation

tf.math.invert\_permutation(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_array\_ops.py.

This operation computes the inverse of an index permutation. It takes a 1-D integer tensor x, which represents the indices of a zero-based array, and swaps each value with its index position. In other words, for an output tensor y and an input tensor x, this operation computes the following:

y[x[i]] = i for i in [0, 1, ..., len(x) - 1]

The values must include 0. There can be no duplicate values or negative values.

#### For example:

# tensor `x` is [3, 4, 0, 2, 1]  
invert\_permutation(x) ==> [2, 4, 3, 0, 1]

#### Args:

* **x**: A Tensor. Must be one of the following types: int32, int64. 1-D.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.in\_top\_k

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/in_top_k#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/in_top_k#aliases)

Says whether the targets are in the top K predictions.

### Aliases:

* tf.compat.v2.math.in\_top\_k
* tf.compat.v2.nn.in\_top\_k
* tf.math.in\_top\_k
* tf.nn.in\_top\_k

tf.math.in\_top\_k(  
    targets,  
    predictions,  
    k,  
    name=None  
)

Defined in [python/ops/nn\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/nn_ops.py).

This outputs a batch\_size bool array, an entry out[i] is true if the prediction for the target class is finite (not inf, -inf, or nan) and among the top k predictions among all predictions for example i. Note that the behavior of InTopK differs from the TopK op in its handling of ties; if multiple classes have the same prediction value and straddle the top-k boundary, all of those classes are considered to be in the top k.

More formally, let

predictionsi be the predictions for all classes for example i, targetsi be the target class for example i, outi be the output for example i,

outi=predictionsi,targetsi∈TopKIncludingTies(predictionsi)

#### Args:

* **predictions**: A Tensor of type float32. A batch\_size x classes tensor.
* **targets**: A Tensor. Must be one of the following types: int32, int64. A batch\_size vector of class ids.
* **k**: An int. Number of top elements to look at for computing precision.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool. Computed Precision at k as a bool Tensor.

# tf.math.is\_finite

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_finite#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_finite#aliases)

Returns which elements of x are finite.

### Aliases:

* tf.compat.v1.debugging.is\_finite
* tf.compat.v1.is\_finite
* tf.compat.v1.math.is\_finite
* tf.compat.v2.math.is\_finite
* tf.math.is\_finite

tf.math.is\_finite(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

#### Numpy Compatibility

Equivalent to np.isfinite

# tf.math.is\_inf

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_inf#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_inf#aliases)

Returns which elements of x are Inf.

### Aliases:

* tf.compat.v1.debugging.is\_inf
* tf.compat.v1.is\_inf
* tf.compat.v1.math.is\_inf
* tf.compat.v2.math.is\_inf
* tf.math.is\_inf

tf.math.is\_inf(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

#### Numpy Compatibility

Equivalent to np.isinf

# tf.math.is\_nan

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_nan#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_nan#aliases)

Returns which elements of x are NaN.

### Aliases:

* tf.compat.v1.debugging.is\_nan
* tf.compat.v1.is\_nan
* tf.compat.v1.math.is\_nan
* tf.compat.v2.math.is\_nan
* tf.math.is\_nan

tf.math.is\_nan(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

#### Numpy Compatibility

Equivalent to np.isnan

# tf.math.is\_non\_decreasing

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_non_decreasing#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_non_decreasing#aliases)

Returns True if x is non-decreasing.

### Aliases:

* tf.compat.v1.debugging.is\_non\_decreasing
* tf.compat.v1.is\_non\_decreasing
* tf.compat.v1.math.is\_non\_decreasing
* tf.compat.v2.math.is\_non\_decreasing
* tf.math.is\_non\_decreasing

tf.math.is\_non\_decreasing(  
    x,  
    name=None  
)

Defined in [python/ops/check\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/check_ops.py).

Elements of x are compared in row-major order. The tensor [x[0],...] is non-decreasing if for every adjacent pair we have x[i] <= x[i+1]. If x has less than two elements, it is trivially non-decreasing.

See also: is\_strictly\_increasing

#### Args:

* **x**: Numeric Tensor.
* **name**: A name for this operation (optional). Defaults to "is\_non\_decreasing"

#### Returns:

Boolean Tensor, equal to True iff x is non-decreasing.

#### Raises:

* **TypeError**: if x is not a numeric tensor.

# tf.math.is\_strictly\_increasing

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_strictly_increasing#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_strictly_increasing#aliases)

Returns True if x is strictly increasing.

### Aliases:

* tf.compat.v1.debugging.is\_strictly\_increasing
* tf.compat.v1.is\_strictly\_increasing
* tf.compat.v1.math.is\_strictly\_increasing
* tf.compat.v2.math.is\_strictly\_increasing
* tf.math.is\_strictly\_increasing

tf.math.is\_strictly\_increasing(  
    x,  
    name=None  
)

Defined in [python/ops/check\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/check_ops.py).

Elements of x are compared in row-major order. The tensor [x[0],...] is strictly increasing if for every adjacent pair we have x[i] < x[i+1]. If x has less than two elements, it is trivially strictly increasing.

See also: is\_non\_decreasing

#### Args:

* **x**: Numeric Tensor.
* **name**: A name for this operation (optional). Defaults to "is\_strictly\_increasing"

#### Returns:

Boolean Tensor, equal to True iff x is strictly increasing.

#### Raises:

* **TypeError**: if x is not a numeric tensor.

# tf.math.l2\_normalize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/l2_normalize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/l2_normalize#aliases)

Normalizes along dimension axis using an L2 norm.

### Aliases:

* tf.compat.v2.linalg.l2\_normalize
* tf.compat.v2.math.l2\_normalize
* tf.compat.v2.nn.l2\_normalize
* tf.linalg.l2\_normalize
* tf.math.l2\_normalize
* tf.nn.l2\_normalize

tf.math.l2\_normalize(  
    x,  
    axis=None,  
    epsilon=1e-12,  
    name=None  
)

Defined in [python/ops/nn\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/nn_impl.py).

For a 1-D tensor with axis = 0, computes

output = x / sqrt(max(sum(x\*\*2), epsilon))

For x with more dimensions, independently normalizes each 1-D slice along dimension axis.

#### Args:

* **x**: A Tensor.
* **axis**: Dimension along which to normalize. A scalar or a vector of integers.
* **epsilon**: A lower bound value for the norm. Will use sqrt(epsilon) as the divisor if norm < sqrt(epsilon).
* **name**: A name for this operation (optional).

#### Returns:

A Tensor with the same shape as x.

# tf.math.lbeta

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/lbeta#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/lbeta#aliases)

Computes ln(|Beta(x)|), reducing along the last dimension.

### Aliases:

* tf.compat.v1.lbeta
* tf.compat.v1.math.lbeta
* tf.compat.v2.math.lbeta
* tf.math.lbeta

tf.math.lbeta(  
    x,  
    name=None  
)

Defined in [python/ops/special\_math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/special_math_ops.py).

Given one-dimensional z = [z\_0,...,z\_{K-1}], we define

Beta(z)=∏jGamma(zj)/Gamma(∑jzj)

And for n + 1 dimensional x with shape [N1, ..., Nn, K], we define

lbeta(x)[i1,...,in]=Log(|Beta(x[i1,...,in,:])|).

In other words, the last dimension is treated as the z vector.

Note that if z = [u, v], then Beta(z)=int01tu−1(1−t)v−1dt, which defines the traditional bivariate beta function.

If the last dimension is empty, we follow the convention that the sum over the empty set is zero, and the product is one.

#### Args:

* **x**: A rank n + 1 Tensor, n >= 0 with type float, or double.
* **name**: A name for the operation (optional).

#### Returns:

The logarithm of |Beta(x)| reducing along the last dimension.

# tf.math.less

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/less#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/less#aliases)

Returns the truth value of (x < y) element-wise.

### Aliases:

* tf.RaggedTensor.\_\_lt\_\_
* tf.Tensor.\_\_lt\_\_
* tf.compat.v1.RaggedTensor.\_\_lt\_\_
* tf.compat.v1.Tensor.\_\_lt\_\_
* tf.compat.v1.less
* tf.compat.v1.math.less
* tf.compat.v2.RaggedTensor.\_\_lt\_\_
* tf.compat.v2.Tensor.\_\_lt\_\_
* tf.compat.v2.less
* tf.compat.v2.math.less
* tf.less
* tf.math.less

tf.math.less(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

NOTE: math.less supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.less\_equal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/less_equal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/less_equal#aliases)

Returns the truth value of (x <= y) element-wise.

### Aliases:

* tf.RaggedTensor.\_\_le\_\_
* tf.Tensor.\_\_le\_\_
* tf.compat.v1.RaggedTensor.\_\_le\_\_
* tf.compat.v1.Tensor.\_\_le\_\_
* tf.compat.v1.less\_equal
* tf.compat.v1.math.less\_equal
* tf.compat.v2.RaggedTensor.\_\_le\_\_
* tf.compat.v2.Tensor.\_\_le\_\_
* tf.compat.v2.less\_equal
* tf.compat.v2.math.less\_equal
* tf.less\_equal
* tf.math.less\_equal

tf.math.less\_equal(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

NOTE: math.less\_equal supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.lgamma

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/lgamma#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/lgamma#aliases)

Computes the log of the absolute value of Gamma(x) element-wise.

### Aliases:

* tf.compat.v1.lgamma
* tf.compat.v1.math.lgamma
* tf.compat.v2.math.lgamma
* tf.math.lgamma

tf.math.lgamma(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.log

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log#used_in_the_tutorials)

Computes natural logarithm of x element-wise.

### Aliases:

* tf.compat.v1.log
* tf.compat.v1.math.log
* tf.compat.v2.math.log
* tf.math.log

tf.math.log(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)

I.e., y=loge⁡x.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.log1p

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log1p#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log1p#aliases)

Computes natural logarithm of (1 + x) element-wise.

### Aliases:

* tf.compat.v1.log1p
* tf.compat.v1.math.log1p
* tf.compat.v2.math.log1p
* tf.math.log1p

tf.math.log1p(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

I.e., y=loge⁡(1+x).

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.logical\_and

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_and#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_and#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_and#used_in_the_tutorials)

Returns the truth value of x AND y element-wise.

### Aliases:

* tf.RaggedTensor.\_\_and\_\_
* tf.compat.v1.RaggedTensor.\_\_and\_\_
* tf.compat.v1.logical\_and
* tf.compat.v1.math.logical\_and
* tf.compat.v2.RaggedTensor.\_\_and\_\_
* tf.compat.v2.logical\_and
* tf.compat.v2.math.logical\_and
* tf.logical\_and
* tf.math.logical\_and

tf.math.logical\_and(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the tutorials:

* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

NOTE: math.logical\_and supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor of type bool.
* **y**: A Tensor of type bool.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.logical\_not

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_not#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_not#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_not#used_in_the_tutorials)

Returns the truth value of NOT x element-wise.

### Aliases:

* tf.RaggedTensor.\_\_invert\_\_
* tf.Tensor.\_\_invert\_\_
* tf.compat.v1.RaggedTensor.\_\_invert\_\_
* tf.compat.v1.Tensor.\_\_invert\_\_
* tf.compat.v1.logical\_not
* tf.compat.v1.math.logical\_not
* tf.compat.v2.RaggedTensor.\_\_invert\_\_
* tf.compat.v2.Tensor.\_\_invert\_\_
* tf.compat.v2.logical\_not
* tf.compat.v2.math.logical\_not
* tf.logical\_not
* tf.math.logical\_not

tf.math.logical\_not(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

#### Args:

* **x**: A Tensor of type bool.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.logical\_or

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_or#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_or#aliases)

Returns the truth value of x OR y element-wise.

### Aliases:

* tf.RaggedTensor.\_\_or\_\_
* tf.compat.v1.RaggedTensor.\_\_or\_\_
* tf.compat.v1.logical\_or
* tf.compat.v1.math.logical\_or
* tf.compat.v2.RaggedTensor.\_\_or\_\_
* tf.compat.v2.logical\_or
* tf.compat.v2.math.logical\_or
* tf.logical\_or
* tf.math.logical\_or

tf.math.logical\_or(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

NOTE: math.logical\_or supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor of type bool.
* **y**: A Tensor of type bool.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.logical\_xor

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_xor#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_xor#aliases)

Logical XOR function.

### Aliases:

* tf.RaggedTensor.\_\_xor\_\_
* tf.compat.v1.RaggedTensor.\_\_xor\_\_
* tf.compat.v1.logical\_xor
* tf.compat.v1.math.logical\_xor
* tf.compat.v2.RaggedTensor.\_\_xor\_\_
* tf.compat.v2.math.logical\_xor
* tf.math.logical\_xor

tf.math.logical\_xor(  
    x,  
    y,  
    name='LogicalXor'  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

x ^ y = (x | y) & ~(x & y)

Inputs are tensor and if the tensors contains more than one element, an element-wise logical XOR is computed.

#### Usage:

x = tf.constant([False, False, True, True], dtype = tf.bool)  
y = tf.constant([False, True, False, True], dtype = tf.bool)  
z = tf.logical\_xor(x, y, name="LogicalXor")  
#  here z = [False  True  True False]

#### Args:

* **x**: A Tensor type bool.
* **y**: A Tensor of type bool.

#### Returns:

A Tensor of type bool with the same size as that of x or y.

# tf.math.log\_sigmoid

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log_sigmoid#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log_sigmoid#aliases)

Computes log sigmoid of x element-wise.

### Aliases:

* tf.compat.v1.log\_sigmoid
* tf.compat.v1.math.log\_sigmoid
* tf.compat.v2.math.log\_sigmoid
* tf.math.log\_sigmoid

tf.math.log\_sigmoid(  
    x,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Specifically, y = log(1 / (1 + exp(-x))). For numerical stability, we use y = -tf.nn.softplus(-x).

#### Args:

* **x**: A Tensor with type float32 or float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor with the same type as x.

# tf.math.maximum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/maximum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/maximum#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/maximum#used_in_the_tutorials)

Returns the max of x and y (i.e. x > y ? x : y) element-wise.

### Aliases:

* tf.compat.v1.math.maximum
* tf.compat.v1.maximum
* tf.compat.v2.math.maximum
* tf.compat.v2.maximum
* tf.math.maximum
* tf.maximum

tf.math.maximum(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the tutorials:

* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

NOTE: math.maximum supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.minimum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/minimum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/minimum#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/minimum#used_in_the_tutorials)

Returns the min of x and y (i.e. x < y ? x : y) element-wise.

### Aliases:

* tf.compat.v1.math.minimum
* tf.compat.v1.minimum
* tf.compat.v2.math.minimum
* tf.compat.v2.minimum
* tf.math.minimum
* tf.minimum

tf.math.minimum(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the tutorials:

* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

NOTE: math.minimum supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.multiply

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/multiply#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/multiply#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/multiply#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/multiply#used_in_the_tutorials)

Returns x \* y element-wise.

### Aliases:

* tf.RaggedTensor.\_\_mul\_\_
* tf.compat.v1.RaggedTensor.\_\_mul\_\_
* tf.compat.v1.math.multiply
* tf.compat.v1.multiply
* tf.compat.v2.RaggedTensor.\_\_mul\_\_
* tf.compat.v2.math.multiply
* tf.compat.v2.multiply
* tf.math.multiply
* tf.multiply

tf.math.multiply(  
    x,  
    y,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

### Used in the guide:

* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

### Used in the tutorials:

* [Automatic differentiation and gradient tape](https://www.tensorflow.org/beta/tutorials/eager/automatic_differentiation)
* [Tensors and Operations](https://www.tensorflow.org/beta/tutorials/eager/basics)

NOTE: <a href="../../tf/math/multiply"><code>tf.multiply</code></a> supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, uint8, int8, uint16, int16, int32, int64, complex64, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.multiply\_no\_nan

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/multiply_no_nan#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/multiply_no_nan#aliases)

Computes the product of x and y and returns 0 if the y is zero, even if x is NaN or infinite.

### Aliases:

* tf.compat.v1.math.multiply\_no\_nan
* tf.compat.v2.math.multiply\_no\_nan
* tf.math.multiply\_no\_nan

tf.math.multiply\_no\_nan(  
    x,  
    y,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64.
* **y**: A Tensor whose dtype is compatible with x.
* **name**: A name for the operation (optional).

#### Returns:

The element-wise value of the x times y.

# tf.math.negative

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/negative#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/negative#aliases)

Computes numerical negative value element-wise.

### Aliases:

* tf.RaggedTensor.\_\_neg\_\_
* tf.Tensor.\_\_neg\_\_
* tf.compat.v1.RaggedTensor.\_\_neg\_\_
* tf.compat.v1.Tensor.\_\_neg\_\_
* tf.compat.v1.math.negative
* tf.compat.v1.negative
* tf.compat.v2.RaggedTensor.\_\_neg\_\_
* tf.compat.v2.Tensor.\_\_neg\_\_
* tf.compat.v2.math.negative
* tf.compat.v2.negative
* tf.math.negative
* tf.negative

tf.math.negative(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

I.e., y=−x.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.negative(x.values, ...), x.dense\_shape)

# tf.math.nextafter

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/nextafter#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/nextafter#aliases)

Returns the next representable value of x1 in the direction of x2, element-wise.

### Aliases:

* tf.compat.v1.math.nextafter
* tf.compat.v2.math.nextafter
* tf.math.nextafter

tf.math.nextafter(  
    x1,  
    x2,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

This operation returns the same result as the C++ std::nextafter function.

It can also return a subnormal number.

#### Args:

* **x1**: A Tensor. Must be one of the following types: float64, float32.
* **x2**: A Tensor. Must have the same type as x1.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x1.

#### Cpp Compatibility

Equivalent to C++ std::nextafter function.

# tf.math.not\_equal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/not_equal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/not_equal#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/not_equal#used_in_the_tutorials)

Returns the truth value of (x != y) element-wise.

### Aliases:

* tf.compat.v1.math.not\_equal
* tf.compat.v1.not\_equal
* tf.compat.v2.math.not\_equal
* tf.compat.v2.not\_equal
* tf.math.not\_equal
* tf.not\_equal

tf.math.not\_equal(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the tutorials:

* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)

NOTE: math.not\_equal supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, uint8, int8, int16, int32, int64, complex64, quint8, qint8, qint32, string, bool, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.polygamma

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/polygamma#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/polygamma#aliases)

Compute the polygamma function ψ(n)(x).

### Aliases:

* tf.compat.v1.math.polygamma
* tf.compat.v1.polygamma
* tf.compat.v2.math.polygamma
* tf.math.polygamma

tf.math.polygamma(  
    a,  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

The polygamma function is defined as:

ψ(a)(x)=dadxaψ(x)

where ψ(x) is the digamma function. The polygamma function is defined only for non-negative integer orders \a\.

#### Args:

* **a**: A Tensor. Must be one of the following types: float32, float64.
* **x**: A Tensor. Must have the same type as a.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as a.

# tf.math.polyval

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/polyval#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/polyval#aliases)

Computes the elementwise value of a polynomial.

### Aliases:

* tf.compat.v1.math.polyval
* tf.compat.v2.math.polyval
* tf.math.polyval

tf.math.polyval(  
    coeffs,  
    x,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

If x is a tensor and coeffs is a list n + 1 tensors, this function returns the value of the n-th order polynomial

p(x) = coeffs[n-1] + coeffs[n-2] \* x + ... + coeffs[0] \* x\*\*(n-1)

evaluated using Horner's method, i.e.

p(x) = coeffs[n-1] + x \* (coeffs[n-2] + ... + x \* (coeffs[1] + x \* coeffs[0]))

#### Args:

* **coeffs**: A list of Tensor representing the coefficients of the polynomial.
* **x**: A Tensor representing the variable of the polynomial.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the shape as the expression p(x) with usual broadcasting rules for element-wise addition and multiplication applied.

#### Numpy Compatibility

Equivalent to numpy.polyval.

# tf.math.pow

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/pow#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/pow#aliases)

Computes the power of one value to another.

### Aliases:

* tf.RaggedTensor.\_\_pow\_\_
* tf.compat.v1.RaggedTensor.\_\_pow\_\_
* tf.compat.v1.math.pow
* tf.compat.v1.pow
* tf.compat.v2.RaggedTensor.\_\_pow\_\_
* tf.compat.v2.math.pow
* tf.compat.v2.pow
* tf.math.pow
* tf.pow

tf.math.pow(  
    x,  
    y,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Given a tensor x and a tensor y, this operation computes xy for corresponding elements in x and y. For example:

x = tf.constant([[2, 2], [3, 3]])  
y = tf.constant([[8, 16], [2, 3]])  
tf.pow(x, y)  # [[256, 65536], [9, 27]]

#### Args:

* **x**: A Tensor of type float16, float32, float64, int32, int64, complex64, or complex128.
* **y**: A Tensor of type float16, float32, float64, int32, int64, complex64, or complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor.

# tf.math.pow

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/pow#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/pow#aliases)

Computes the power of one value to another.

### Aliases:

* tf.RaggedTensor.\_\_pow\_\_
* tf.compat.v1.RaggedTensor.\_\_pow\_\_
* tf.compat.v1.math.pow
* tf.compat.v1.pow
* tf.compat.v2.RaggedTensor.\_\_pow\_\_
* tf.compat.v2.math.pow
* tf.compat.v2.pow
* tf.math.pow
* tf.pow

tf.math.pow(  
    x,  
    y,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Given a tensor x and a tensor y, this operation computes xy for corresponding elements in x and y. For example:

x = tf.constant([[2, 2], [3, 3]])  
y = tf.constant([[8, 16], [2, 3]])  
tf.pow(x, y)  # [[256, 65536], [9, 27]]

#### Args:

* **x**: A Tensor of type float16, float32, float64, int32, int64, complex64, or complex128.
* **y**: A Tensor of type float16, float32, float64, int32, int64, complex64, or complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor.

# tf.math.reciprocal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reciprocal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reciprocal#aliases)

Computes the reciprocal of x element-wise.

### Aliases:

* tf.compat.v1.math.reciprocal
* tf.compat.v1.reciprocal
* tf.compat.v2.math.reciprocal
* tf.math.reciprocal

tf.math.reciprocal(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

I.e., y=1/x.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.reduce\_any

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_any#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_any#aliases)

Computes the "logical or" of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_any
* tf.compat.v2.reduce\_any
* tf.math.reduce\_any
* tf.reduce\_any

tf.math.reduce\_any(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[True,  True], [False, False]])  
tf.reduce\_any(x)  # True  
tf.reduce\_any(x, 0)  # [True, True]  
tf.reduce\_any(x, 1)  # [True, False]

#### Args:

* **input\_tensor**: The boolean tensor to reduce.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

#### Numpy Compatibility

Equivalent to np.any

# tf.math.reduce\_euclidean\_norm

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_euclidean_norm#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_euclidean_norm#aliases)

Computes the Euclidean norm of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v1.math.reduce\_euclidean\_norm
* tf.compat.v2.math.reduce\_euclidean\_norm
* tf.math.reduce\_euclidean\_norm

tf.math.reduce\_euclidean\_norm(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[1, 2, 3], [1, 1, 1]])  
tf.reduce\_euclidean\_norm(x)  # sqrt(17)  
tf.reduce\_euclidean\_norm(x, 0)  # [sqrt(2), sqrt(5), sqrt(10)]  
tf.reduce\_euclidean\_norm(x, 1)  # [sqrt(14), sqrt(3)]  
tf.reduce\_euclidean\_norm(x, 1, keepdims=True)  # [[sqrt(14)], [sqrt(3)]]  
tf.reduce\_euclidean\_norm(x, [0, 1])  # sqrt(17)

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor, of the same dtype as the input\_tensor.

# tf.math.reduce\_logsumexp

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_logsumexp#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_logsumexp#aliases)

Computes log(sum(exp(elements across dimensions of a tensor))).

### Aliases:

* tf.compat.v2.math.reduce\_logsumexp
* tf.compat.v2.reduce\_logsumexp
* tf.math.reduce\_logsumexp
* tf.reduce\_logsumexp

tf.math.reduce\_logsumexp(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis has no entries, all dimensions are reduced, and a tensor with a single element is returned.

This function is more numerically stable than log(sum(exp(input))). It avoids overflows caused by taking the exp of large inputs and underflows caused by taking the log of small inputs.

#### For example:

x = tf.constant([[0., 0., 0.], [0., 0., 0.]])  
tf.reduce\_logsumexp(x)  # log(6)  
tf.reduce\_logsumexp(x, 0)  # [log(2), log(2), log(2)]  
tf.reduce\_logsumexp(x, 1)  # [log(3), log(3)]  
tf.reduce\_logsumexp(x, 1, keepdims=True)  # [[log(3)], [log(3)]]  
tf.reduce\_logsumexp(x, [0, 1])  # log(6)

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

# tf.math.reduce\_max

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_max#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_max#aliases)

Computes the maximum of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_max
* tf.compat.v2.reduce\_max
* tf.math.reduce\_max
* tf.reduce\_max

tf.math.reduce\_max(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### Args:

* **input\_tensor**: The tensor to reduce. Should have real numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

#### Numpy Compatibility

Equivalent to np.max

# tf.math.reduce\_mean

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean#used_in_the_tutorials)

Computes the mean of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_mean
* tf.compat.v2.reduce\_mean
* tf.math.reduce\_mean
* tf.reduce\_mean

tf.math.reduce\_mean(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)
* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Custom training: basics](https://www.tensorflow.org/beta/tutorials/eager/custom_training)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[1., 1.], [2., 2.]])  
tf.reduce\_mean(x)  # 1.5  
tf.reduce\_mean(x, 0)  # [1.5, 1.5]  
tf.reduce\_mean(x, 1)  # [1.,  2.]

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

#### Numpy Compatibility

Equivalent to np.mean

Please note that np.mean has a dtype parameter that could be used to specify the output type. By default this is dtype=float64. On the other hand, [tf.reduce\_mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean) has an aggressive type inference from input\_tensor, for example:

x = tf.constant([1, 0, 1, 0])  
tf.reduce\_mean(x)  # 0  
y = tf.constant([1., 0., 1., 0.])  
tf.reduce\_mean(y)  # 0.5

# tf.math.reduce\_min

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_min#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_min#aliases)

Computes the minimum of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_min
* tf.compat.v2.reduce\_min
* tf.math.reduce\_min
* tf.reduce\_min

tf.math.reduce\_min(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### Args:

* **input\_tensor**: The tensor to reduce. Should have real numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

#### Numpy Compatibility

Equivalent to np.min

# tf.math.reduce\_prod

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_prod#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_prod#aliases)

Computes the product of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_prod
* tf.compat.v2.reduce\_prod
* tf.math.reduce\_prod
* tf.reduce\_prod

tf.math.reduce\_prod(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

#### Numpy Compatibility

Equivalent to np.prod

# tf.math.reduce\_std

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_std#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_std#aliases)

Computes the standard deviation of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v1.math.reduce\_std
* tf.compat.v2.math.reduce\_std
* tf.math.reduce\_std

tf.math.reduce\_std(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[1., 2.], [3., 4.]])  
tf.reduce\_std(x)  # 1.1180339887498949  
tf.reduce\_std(x, 0)  # [1., 1.]  
tf.reduce\_std(x, 1)  # [0.5,  0.5]

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name scope for the associated operations (optional).

#### Returns:

The reduced tensor, of the same dtype as the input\_tensor.

#### Numpy Compatibility

Equivalent to np.std

Please note that np.std has a dtype parameter that could be used to specify the output type. By default this is dtype=float64. On the other hand, tf.reduce\_std has an aggressive type inference from input\_tensor,

# tf.math.reduce\_sum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_sum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_sum#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_sum#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_sum#used_in_the_tutorials)

Computes the sum of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_sum
* tf.compat.v2.reduce\_sum
* tf.math.reduce\_sum
* tf.reduce\_sum

tf.math.reduce\_sum(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

### Used in the guide:

* [Distributed training in TensorFlow](https://www.tensorflow.org/beta/guide/distribute_strategy)
* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

### Used in the tutorials:

* [Automatic differentiation and gradient tape](https://www.tensorflow.org/beta/tutorials/eager/automatic_differentiation)
* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Multi-worker Training with Estimator](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_estimator)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Tensors and Operations](https://www.tensorflow.org/beta/tutorials/eager/basics)
* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)
* [tf.function](https://www.tensorflow.org/beta/tutorials/eager/tf_function)

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[1, 1, 1], [1, 1, 1]])  
tf.reduce\_sum(x)  # 6  
tf.reduce\_sum(x, 0)  # [2, 2, 2]  
tf.reduce\_sum(x, 1)  # [3, 3]  
tf.reduce\_sum(x, 1, keepdims=True)  # [[3], [3]]  
tf.reduce\_sum(x, [0, 1])  # 6

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor, of the same dtype as the input\_tensor.

#### Numpy Compatibility

Equivalent to np.sum apart the fact that numpy upcast uint8 and int32 to int64 while tensorflow returns the same dtype as the input.

# tf.math.reduce\_variance

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_variance#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_variance#aliases)

Computes the variance of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v1.math.reduce\_variance
* tf.compat.v2.math.reduce\_variance
* tf.math.reduce\_variance

tf.math.reduce\_variance(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[1., 2.], [3., 4.]])  
tf.reduce\_variance(x)  # 1.25  
tf.reduce\_variance(x, 0)  # [1., 1.]  
tf.reduce\_variance(x, 1)  # [0.25,  0.25]

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name scope for the associated operations (optional).

#### Returns:

The reduced tensor, of the same dtype as the input\_tensor.

#### Numpy Compatibility

Equivalent to np.var

Please note that np.var has a dtype parameter that could be used to specify the output type. By default this is dtype=float64. On the other hand, tf.reduce\_variance has an aggressive type inference from input\_tensor,

# tf.math.rint

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/rint#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/rint#aliases)

Returns element-wise integer closest to x.

### Aliases:

* tf.compat.v1.math.rint
* tf.compat.v1.rint
* tf.compat.v2.math.rint
* tf.math.rint

tf.math.rint(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

If the result is midway between two representable values, the even representable is chosen. For example:

rint(-1.5) ==> -2.0  
rint(0.5000001) ==> 1.0  
rint([-1.7, -1.5, -0.2, 0.2, 1.5, 1.7, 2.0]) ==> [-2., -2., -0., 0., 2., 2., 2.]

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.round

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/round#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/round#aliases)

Rounds the values of a tensor to the nearest integer, element-wise.

### Aliases:

* tf.compat.v1.math.round
* tf.compat.v1.round
* tf.compat.v2.math.round
* tf.compat.v2.round
* tf.math.round
* tf.round

tf.math.round(  
    x,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Rounds half to even. Also known as bankers rounding. If you want to round according to the current system rounding mode use tf::cint. For example:

x = tf.constant([0.9, 2.5, 2.3, 1.5, -4.5])  
tf.round(x)  # [ 1.0, 2.0, 2.0, 2.0, -4.0 ]

#### Args:

* **x**: A Tensor of type float16, float32, float64, int32, or int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of same shape and type as x.

# tf.math.rsqrt

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/rsqrt#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/rsqrt#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/rsqrt#used_in_the_tutorials)

Computes reciprocal of square root of x element-wise.

### Aliases:

* tf.compat.v1.math.rsqrt
* tf.compat.v1.rsqrt
* tf.compat.v2.math.rsqrt
* tf.math.rsqrt

tf.math.rsqrt(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the tutorials:

* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

I.e., y=1/x.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.scalar\_mul

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/scalar_mul#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/scalar_mul#aliases)

Multiplies a scalar times a Tensor or IndexedSlices object.

### Aliases:

* tf.compat.v2.math.scalar\_mul
* tf.compat.v2.scalar\_mul
* tf.math.scalar\_mul
* tf.scalar\_mul

tf.math.scalar\_mul(  
    scalar,  
    x,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Intended for use in gradient code which might deal with IndexedSlices objects, which are easy to multiply by a scalar but more expensive to multiply with arbitrary tensors.

#### Args:

* **scalar**: A 0-D scalar Tensor. Must have known shape.
* **x**: A Tensor or IndexedSlices to be scaled.
* **name**: A name for the operation (optional).

#### Returns:

scalar \* x of the same type (Tensor or IndexedSlices) as x.

#### Raises:

* **ValueError**: if scalar is not a 0-D scalar.

# tf.math.segment\_max

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_max#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_max#aliases)

Computes the maximum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.segment\_max
* tf.compat.v1.segment\_max
* tf.compat.v2.math.segment\_max
* tf.math.segment\_max

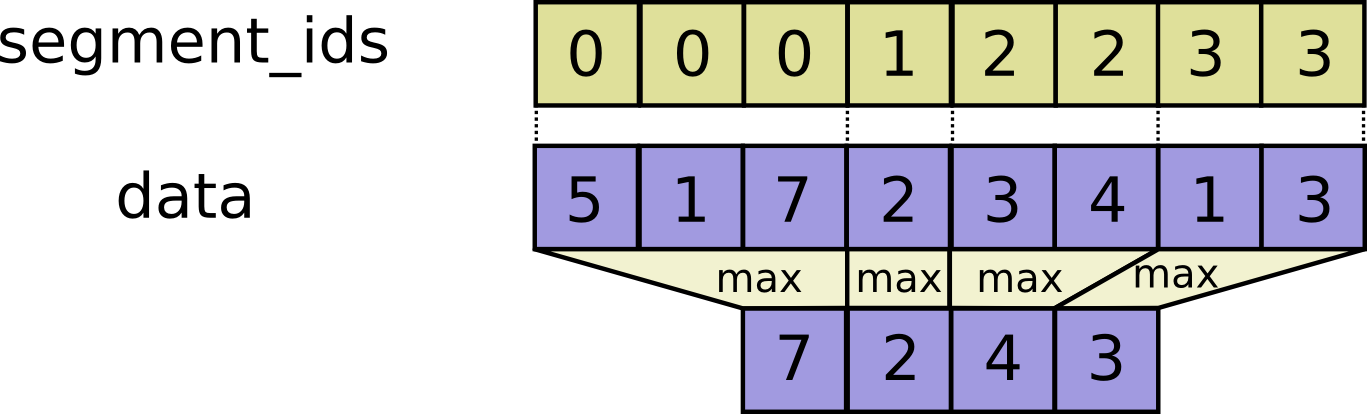
tf.math.segment\_max(  
    data,  
    segment\_ids,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that outputi=maxj(dataj) where max is over j such that segment\_ids[j] == i.

If the max is empty for a given segment ID i, output[i] = 0.



#### For example:

c = tf.constant([[1,2,3,4], [4, 3, 2, 1], [5,6,7,8]])  
tf.segment\_max(c, tf.constant([0, 0, 1]))  
# ==> [[4, 3, 3, 4],  
#      [5, 6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A 1-D tensor whose size is equal to the size of data's first dimension. Values should be sorted and can be repeated.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.segment\_mean

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_mean#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_mean#aliases)

Computes the mean along segments of a tensor.

### Aliases:

* tf.compat.v1.math.segment\_mean
* tf.compat.v1.segment\_mean
* tf.compat.v2.math.segment\_mean
* tf.math.segment\_mean

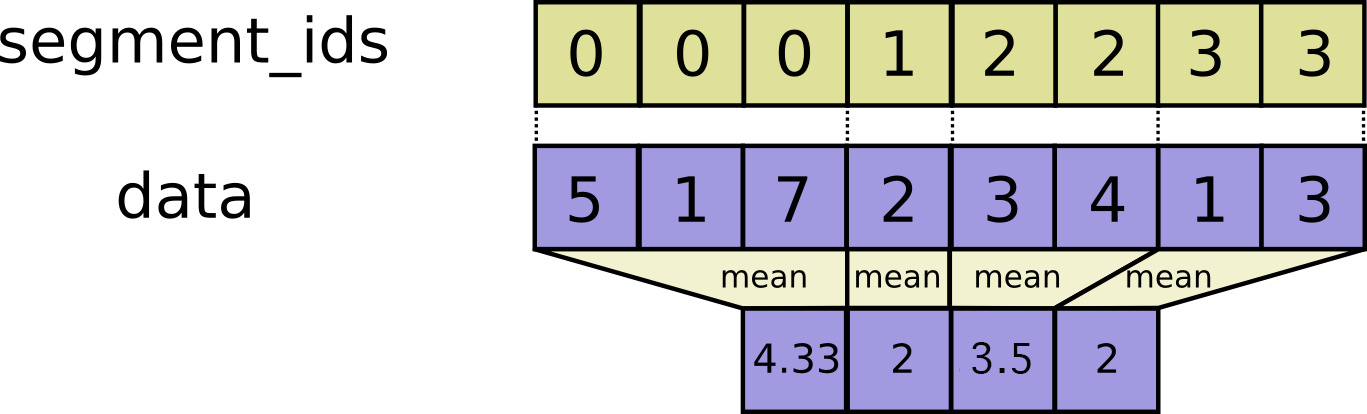
tf.math.segment\_mean(  
    data,  
    segment\_ids,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that outputi=∑jdatajN where mean is over j such that segment\_ids[j] == i and N is the total number of values summed.

If the mean is empty for a given segment ID i, output[i] = 0.



#### For example:

c = tf.constant([[1.0,2,3,4], [4, 3, 2, 1], [5,6,7,8]])  
tf.segment\_mean(c, tf.constant([0, 0, 1]))  
# ==> [[2.5, 2.5, 2.5, 2.5],  
#      [5, 6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A 1-D tensor whose size is equal to the size of data's first dimension. Values should be sorted and can be repeated.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.segment\_min

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_min#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_min#aliases)

Computes the minimum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.segment\_min
* tf.compat.v1.segment\_min
* tf.compat.v2.math.segment\_min
* tf.math.segment\_min

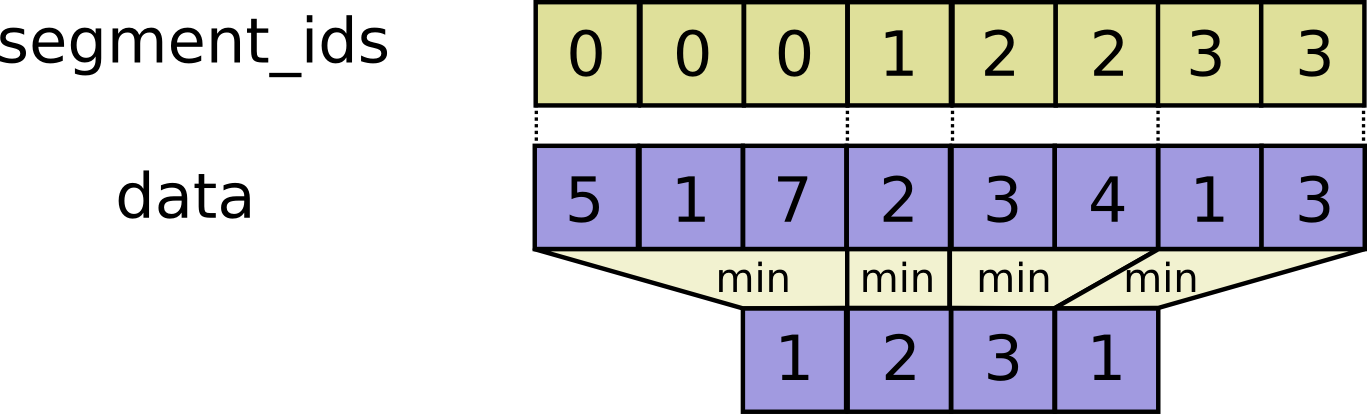
tf.math.segment\_min(  
    data,  
    segment\_ids,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that outputi=minj(dataj) where min is over j such that segment\_ids[j] == i.

If the min is empty for a given segment ID i, output[i] = 0.



#### For example:

c = tf.constant([[1,2,3,4], [4, 3, 2, 1], [5,6,7,8]])  
tf.segment\_min(c, tf.constant([0, 0, 1]))  
# ==> [[1, 2, 2, 1],  
#      [5, 6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A 1-D tensor whose size is equal to the size of data's first dimension. Values should be sorted and can be repeated.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.segment\_prod

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_prod#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_prod#aliases)

Computes the product along segments of a tensor.

### Aliases:

* tf.compat.v1.math.segment\_prod
* tf.compat.v1.segment\_prod
* tf.compat.v2.math.segment\_prod
* tf.math.segment\_prod

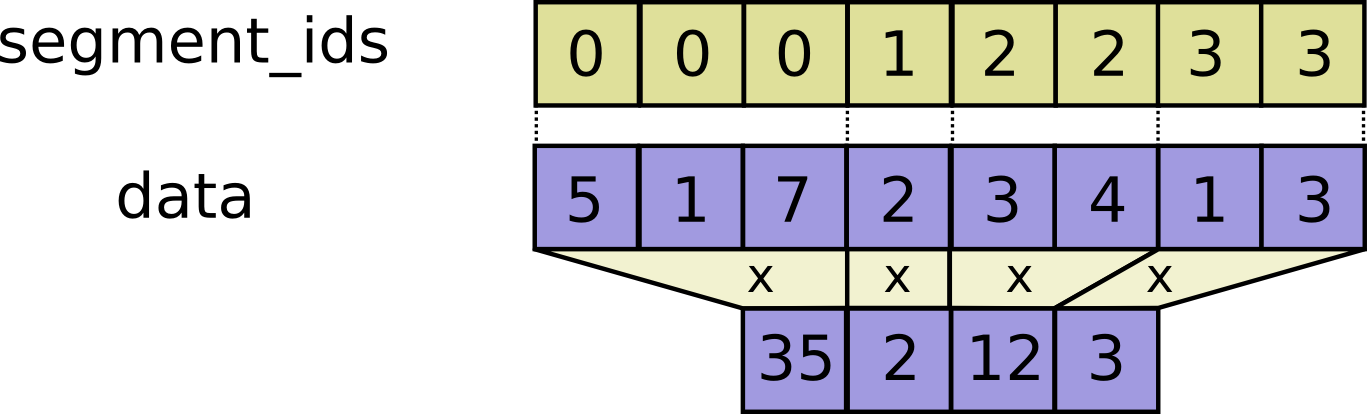
tf.math.segment\_prod(  
    data,  
    segment\_ids,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that outputi=∏jdataj where the product is over j such that segment\_ids[j] == i.

If the product is empty for a given segment ID i, output[i] = 1.



#### For example:

c = tf.constant([[1,2,3,4], [4, 3, 2, 1], [5,6,7,8]])  
tf.segment\_prod(c, tf.constant([0, 0, 1]))  
# ==> [[4, 6, 6, 4],  
#      [5, 6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A 1-D tensor whose size is equal to the size of data's first dimension. Values should be sorted and can be repeated.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.segment\_sum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_sum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_sum#aliases)

Computes the sum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.segment\_sum
* tf.compat.v1.segment\_sum
* tf.compat.v2.math.segment\_sum
* tf.math.segment\_sum

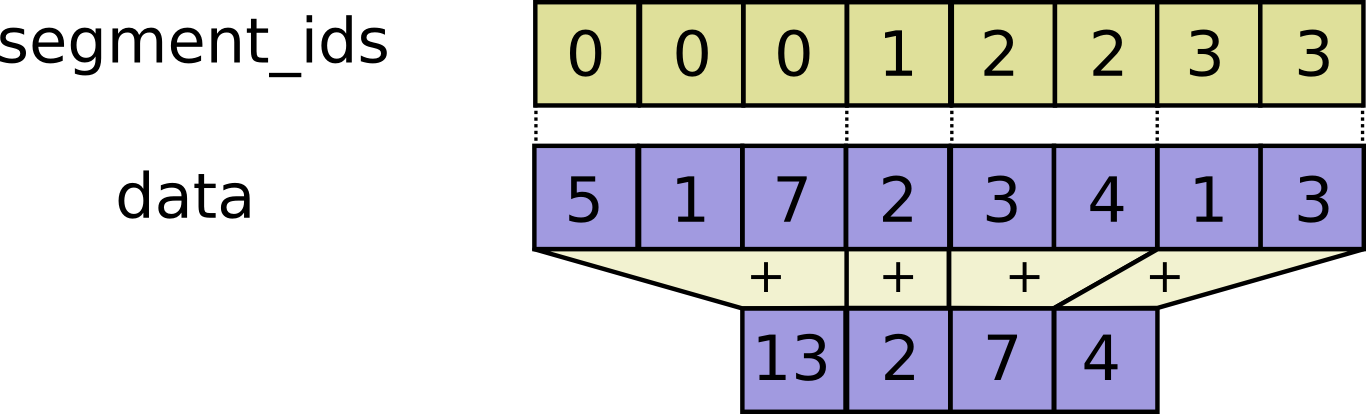
tf.math.segment\_sum(  
    data,  
    segment\_ids,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that outputi=∑jdataj where sum is over j such that segment\_ids[j] == i.

If the sum is empty for a given segment ID i, output[i] = 0.



#### For example:

c = tf.constant([[1,2,3,4], [4, 3, 2, 1], [5,6,7,8]])  
tf.segment\_sum(c, tf.constant([0, 0, 1]))  
# ==> [[5, 5, 5, 5],  
#      [5, 6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A 1-D tensor whose size is equal to the size of data's first dimension. Values should be sorted and can be repeated.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.sigmoid

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sigmoid#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sigmoid#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sigmoid#used_in_the_tutorials)

Computes sigmoid of x element-wise.

### Aliases:

* tf.compat.v1.math.sigmoid
* tf.compat.v1.nn.sigmoid
* tf.compat.v1.sigmoid
* tf.compat.v2.math.sigmoid
* tf.compat.v2.nn.sigmoid
* tf.compat.v2.sigmoid
* tf.math.sigmoid
* tf.nn.sigmoid
* tf.sigmoid

tf.math.sigmoid(  
    x,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)

Specifically, y = 1 / (1 + exp(-x)).

#### Args:

* **x**: A Tensor with type float16, float32, float64, complex64, or complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor with the same type as x.

#### Scipy Compatibility

Equivalent to scipy.special.expit

# tf.math.sign

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sign#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sign#aliases)

Returns an element-wise indication of the sign of a number.

### Aliases:

* tf.compat.v1.math.sign
* tf.compat.v1.sign
* tf.compat.v2.math.sign
* tf.compat.v2.sign
* tf.math.sign
* tf.sign

tf.math.sign(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

y = sign(x) = -1 if x < 0; 0 if x == 0; 1 if x > 0.

For complex numbers, y = sign(x) = x / |x| if x != 0, otherwise y = 0.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.sign(x.values, ...), x.dense\_shape)

# tf.math.sin

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sin#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sin#aliases)

Computes sin of x element-wise.

### Aliases:

* tf.compat.v1.math.sin
* tf.compat.v1.sin
* tf.compat.v2.math.sin
* tf.compat.v2.sin
* tf.math.sin
* tf.sin

tf.math.sin(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.sinh

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sinh#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sinh#aliases)

Computes hyperbolic sine of x element-wise.

### Aliases:

* tf.compat.v1.math.sinh
* tf.compat.v1.sinh
* tf.compat.v2.math.sinh
* tf.compat.v2.sinh
* tf.math.sinh
* tf.sinh

tf.math.sinh(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.softplus

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/softplus#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/softplus#aliases)

Computes softplus: log(exp(features) + 1).

### Aliases:

* tf.compat.v1.math.softplus
* tf.compat.v1.nn.softplus
* tf.compat.v2.math.softplus
* tf.compat.v2.nn.softplus
* tf.math.softplus
* tf.nn.softplus

tf.math.softplus(  
    features,  
    name=None  
)

Defined in generated file: python/ops/gen\_nn\_ops.py.

#### Args:

* **features**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as features.

# tf.math.sqrt

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sqrt#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sqrt#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sqrt#used_in_the_tutorials)

Computes square root of x element-wise.

### Aliases:

* tf.compat.v1.math.sqrt
* tf.compat.v1.sqrt
* tf.compat.v2.math.sqrt
* tf.compat.v2.sqrt
* tf.math.sqrt
* tf.sqrt

tf.math.sqrt(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the tutorials:

* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

I.e., y=x=x1/2.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.sqrt(x.values, ...), x.dense\_shape)

# tf.math.square

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/square#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/square#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/square#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/square#used_in_the_tutorials)

Computes square of x element-wise.

### Aliases:

* tf.compat.v1.math.square
* tf.compat.v1.square
* tf.compat.v2.math.square
* tf.compat.v2.square
* tf.math.square
* tf.square

tf.math.square(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

### Used in the tutorials:

* [Custom training: basics](https://www.tensorflow.org/beta/tutorials/eager/custom_training)
* [Tensors and Operations](https://www.tensorflow.org/beta/tutorials/eager/basics)

I.e., y=x∗x=x2.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.square(x.values, ...), x.dense\_shape)

# tf.math.squared\_difference

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/squared_difference#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/squared_difference#aliases)

Returns (x - y)(x - y) element-wise.

### Aliases:

* tf.compat.v1.math.squared\_difference
* tf.compat.v1.squared\_difference
* tf.compat.v2.math.squared\_difference
* tf.math.squared\_difference

tf.math.squared\_difference(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

NOTE: math.squared\_difference supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.subtract

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/subtract#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/subtract#aliases)

Returns x - y element-wise.

### Aliases:

* tf.RaggedTensor.\_\_sub\_\_
* tf.compat.v1.RaggedTensor.\_\_sub\_\_
* tf.compat.v1.math.subtract
* tf.compat.v1.subtract
* tf.compat.v2.RaggedTensor.\_\_sub\_\_
* tf.compat.v2.math.subtract
* tf.compat.v2.subtract
* tf.math.subtract
* tf.subtract

tf.math.subtract(  
    x,  
    y,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

NOTE: Subtract supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, uint8, int8, uint16, int16, int32, int64, complex64, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.tan

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tan#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tan#aliases)

Computes tan of x element-wise.

### Aliases:

* tf.compat.v1.math.tan
* tf.compat.v1.tan
* tf.compat.v2.math.tan
* tf.compat.v2.tan
* tf.math.tan
* tf.tan

tf.math.tan(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.tanh

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tanh#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tanh#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tanh#used_in_the_tutorials)

Computes hyperbolic tangent of x element-wise.

### Aliases:

* tf.compat.v1.math.tanh
* tf.compat.v1.nn.tanh
* tf.compat.v1.tanh
* tf.compat.v2.math.tanh
* tf.compat.v2.nn.tanh
* tf.compat.v2.tanh
* tf.math.tanh
* tf.nn.tanh
* tf.tanh

tf.math.tanh(  
    x,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [tf.function](https://www.tensorflow.org/beta/tutorials/eager/tf_function)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.tanh(x.values, ...), x.dense\_shape)

# tf.math.top\_k

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/top_k#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/top_k#aliases)

Finds values and indices of the k largest entries for the last dimension.

### Aliases:

* tf.compat.v1.math.top\_k
* tf.compat.v1.nn.top\_k
* tf.compat.v2.math.top\_k
* tf.compat.v2.nn.top\_k
* tf.math.top\_k
* tf.nn.top\_k

tf.math.top\_k(  
    input,  
    k=1,  
    sorted=True,  
    name=None  
)

Defined in [python/ops/nn\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/nn_ops.py).

If the input is a vector (rank=1), finds the k largest entries in the vector and outputs their values and indices as vectors. Thus values[j] is the j-th largest entry in input, and its index is indices[j].

For matrices (resp. higher rank input), computes the top k entries in each row (resp. vector along the last dimension). Thus,

values.shape = indices.shape = input.shape[:-1] + [k]

If two elements are equal, the lower-index element appears first.

#### Args:

* **input**: 1-D or higher Tensor with last dimension at least k.
* **k**: 0-D int32 Tensor. Number of top elements to look for along the last dimension (along each row for matrices).
* **sorted**: If true the resulting k elements will be sorted by the values in descending order.
* **name**: Optional name for the operation.

#### Returns:

* **values**: The k largest elements along each last dimensional slice.
* **indices**: The indices of values within the last dimension of input.

# tf.math.truediv

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/truediv#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/truediv#aliases)

Divides x / y elementwise (using Python 3 division operator semantics).

### Aliases:

* tf.RaggedTensor.\_\_truediv\_\_
* tf.compat.v1.RaggedTensor.\_\_truediv\_\_
* tf.compat.v1.math.truediv
* tf.compat.v1.truediv
* tf.compat.v2.RaggedTensor.\_\_truediv\_\_
* tf.compat.v2.math.truediv
* tf.compat.v2.truediv
* tf.math.truediv
* tf.truediv

tf.math.truediv(  
    x,  
    y,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

NOTE: Prefer using the Tensor operator or tf.divide which obey Python division operator semantics.

This function forces Python 3 division operator semantics where all integer arguments are cast to floating types first. This op is generated by normal x / y division in Python 3 and in Python 2.7 withfrom \_\_future\_\_ import division. If you want integer division that rounds down, use x // y or [tf.math.floordiv](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floordiv).

x and y must have the same numeric type. If the inputs are floating point, the output will have the same type. If the inputs are integral, the inputs are cast to float32 for int8 and int16 and float64 for int32 and int64 (matching the behavior of Numpy).

#### Args:

* **x**: Tensor numerator of numeric type.
* **y**: Tensor denominator of numeric type.
* **name**: A name for the operation (optional).

#### Returns:

x / y evaluated in floating point.

#### Raises:

* **TypeError**: If x and y have different dtypes.

# tf.math.unsorted\_segment\_max

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_max#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_max#aliases)

Computes the maximum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_max
* tf.compat.v1.unsorted\_segment\_max
* tf.compat.v2.math.unsorted\_segment\_max
* tf.math.unsorted\_segment\_max

tf.math.unsorted\_segment\_max(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

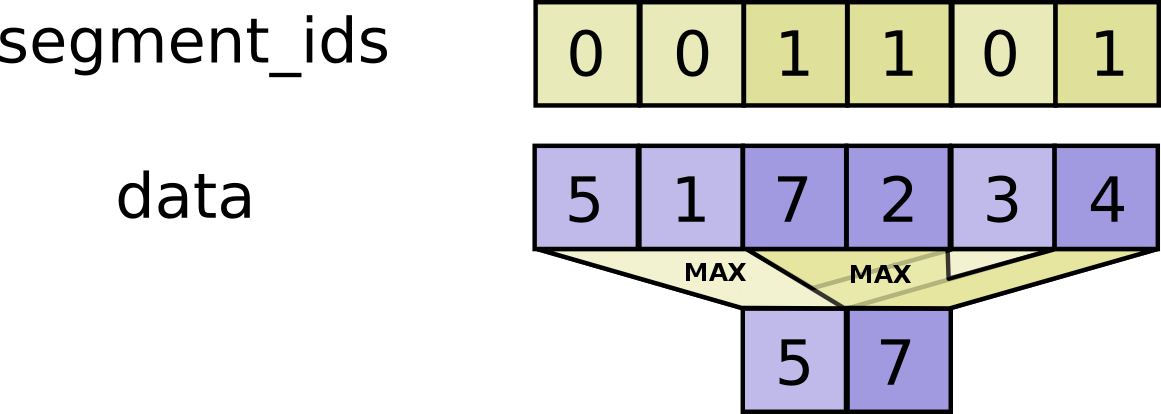
Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

This operator is similar to the unsorted segment sum operator found [(here)](https://www.tensorflow.org/versions/r2.0/api_docs/api_docs/python/math_ops#UnsortedSegmentSum). Instead of computing the sum over segments, it computes the maximum such that:

outputi=maxj...data[j...] where max is over tuples j... such that segment\_ids[j...] == i.

If the maximum is empty for a given segment ID i, it outputs the smallest possible value for the specific numeric type, output[i] = numeric\_limits<T>::lowest().

If the given segment ID i is negative, then the corresponding value is dropped, and will not be included in the result.



#### For example:

c = tf.constant([[1,2,3,4], [5,6,7,8], [4,3,2,1]])  
tf.unsorted\_segment\_max(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 4,  3, 3, 4],  
#       [5,  6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A tensor whose shape is a prefix of data.shape.
* **num\_segments**: A Tensor. Must be one of the following types: int32, int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.unsorted\_segment\_mean

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_mean#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_mean#aliases)

Computes the mean along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_mean
* tf.compat.v1.unsorted\_segment\_mean
* tf.compat.v2.math.unsorted\_segment\_mean
* tf.math.unsorted\_segment\_mean

tf.math.unsorted\_segment\_mean(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

Defined in [python/ops/math\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/math_ops.py).

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

This operator is similar to the unsorted segment sum operator found [here](https://www.tensorflow.org/versions/r2.0/api_docs/api_docs/python/math_ops#UnsortedSegmentSum). Instead of computing the sum over segments, it computes the mean of all entries belonging to a segment such that:

outputi=1/Ni∑j...data[j...] where the sum is over tuples j... such that segment\_ids[j...] == i with \N\_i\ being the number of occurrences of id \i\.

If there is no entry for a given segment ID i, it outputs 0.

If the given segment ID i is negative, the value is dropped and will not be added to the sum of the segment.

#### Args:

* **data**: A Tensor with floating point or complex dtype.
* **segment\_ids**: An integer tensor whose shape is a prefix of data.shape.
* **num\_segments**: An integer scalar Tensor. The number of distinct segment IDs.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has same shape as data, except for the first segment\_ids.rank dimensions, which are replaced with a single dimension which has size num\_segments.

# tf.math.unsorted\_segment\_min

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_min#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_min#aliases)

Computes the minimum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_min
* tf.compat.v1.unsorted\_segment\_min
* tf.compat.v2.math.unsorted\_segment\_min
* tf.math.unsorted\_segment\_min

tf.math.unsorted\_segment\_min(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

This operator is similar to the unsorted segment sum operator found [(here)](https://www.tensorflow.org/versions/r2.0/api_docs/api_docs/python/math_ops#UnsortedSegmentSum). Instead of computing the sum over segments, it computes the minimum such that:

outputi=minj...data[j...] where min is over tuples j... such that segment\_ids[j...] == i.

If the minimum is empty for a given segment ID i, it outputs the largest possible value for the specific numeric type, output[i] = numeric\_limits<T>::max().

#### For example:

c = tf.constant([[1,2,3,4], [5,6,7,8], [4,3,2,1]])  
tf.unsorted\_segment\_min(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 1,  2, 2, 1],  
#       [5,  6, 7, 8]]

If the given segment ID i is negative, then the corresponding value is dropped, and will not be included in the result.

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A tensor whose shape is a prefix of data.shape.
* **num\_segments**: A Tensor. Must be one of the following types: int32, int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.unsorted\_segment\_prod

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_prod#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_prod#aliases)

Computes the product along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_prod
* tf.compat.v1.unsorted\_segment\_prod
* tf.compat.v2.math.unsorted\_segment\_prod
* tf.math.unsorted\_segment\_prod

tf.math.unsorted\_segment\_prod(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

This operator is similar to the unsorted segment sum operator found [(here)](https://www.tensorflow.org/versions/r2.0/api_docs/api_docs/python/math_ops#UnsortedSegmentSum). Instead of computing the sum over segments, it computes the product of all entries belonging to a segment such that:

outputi=∏j...data[j...] where the product is over tuples j... such that segment\_ids[j...] == i.

#### For example:

c = tf.constant([[1,2,3,4], [5,6,7,8], [4,3,2,1]])  
tf.unsorted\_segment\_prod(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 4,  6, 6, 4],  
#       [5,  6, 7, 8]]

If there is no entry for a given segment ID i, it outputs 1.

If the given segment ID i is negative, then the corresponding value is dropped, and will not be included in the result.

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A tensor whose shape is a prefix of data.shape.
* **num\_segments**: A Tensor. Must be one of the following types: int32, int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.unsorted\_segment\_prod

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_prod#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_prod#aliases)

Computes the product along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_prod
* tf.compat.v1.unsorted\_segment\_prod
* tf.compat.v2.math.unsorted\_segment\_prod
* tf.math.unsorted\_segment\_prod

tf.math.unsorted\_segment\_prod(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

This operator is similar to the unsorted segment sum operator found [(here)](https://www.tensorflow.org/versions/r2.0/api_docs/api_docs/python/math_ops#UnsortedSegmentSum). Instead of computing the sum over segments, it computes the product of all entries belonging to a segment such that:

outputi=∏j...data[j...] where the product is over tuples j... such that segment\_ids[j...] == i.

#### For example:

c = tf.constant([[1,2,3,4], [5,6,7,8], [4,3,2,1]])  
tf.unsorted\_segment\_prod(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 4,  6, 6, 4],  
#       [5,  6, 7, 8]]

If there is no entry for a given segment ID i, it outputs 1.

If the given segment ID i is negative, then the corresponding value is dropped, and will not be included in the result.

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A tensor whose shape is a prefix of data.shape.
* **num\_segments**: A Tensor. Must be one of the following types: int32, int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.unsorted\_segment\_sum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_sum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_sum#aliases)

Computes the sum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_sum
* tf.compat.v1.unsorted\_segment\_sum
* tf.compat.v2.math.unsorted\_segment\_sum
* tf.math.unsorted\_segment\_sum

tf.math.unsorted\_segment\_sum(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

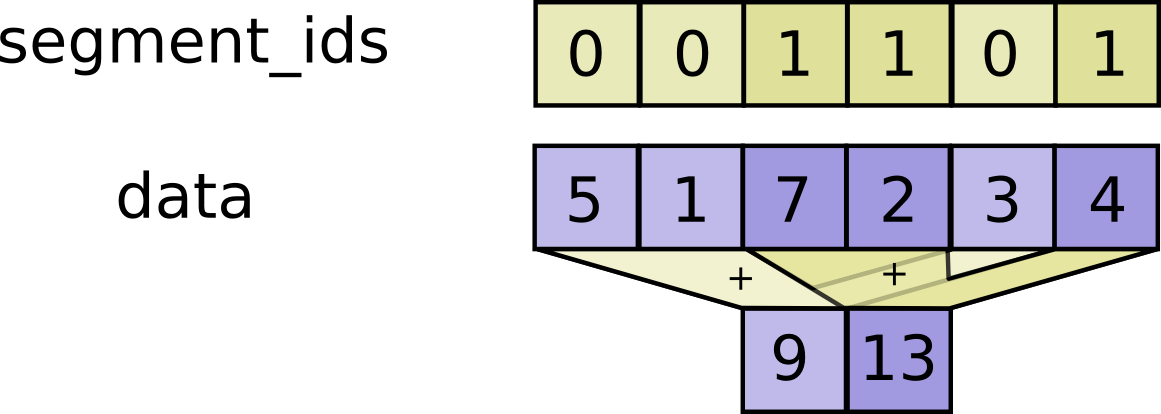
Defined in generated file: python/ops/gen\_math\_ops.py.

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that output[i]=∑j...data[j...] where the sum is over tuples j... such that segment\_ids[j...] == i. Unlike SegmentSum, segment\_ids need not be sorted and need not cover all values in the full range of valid values.

If the sum is empty for a given segment ID i, output[i] = 0. If the given segment ID i is negative, the value is dropped and will not be added to the sum of the segment.

num\_segments should equal the number of distinct segment IDs.



c = tf.constant([[1,2,3,4], [5,6,7,8], [4,3,2,1]])  
tf.unsorted\_segment\_sum(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 5,  5, 5, 5],  
#       [5,  6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A tensor whose shape is a prefix of data.shape.
* **num\_segments**: A Tensor. Must be one of the following types: int32, int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.xdivy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/xdivy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/xdivy#aliases)

Returns 0 if x == 0, and x / y otherwise, elementwise.

### Aliases:

* tf.compat.v1.math.xdivy
* tf.compat.v2.math.xdivy
* tf.math.xdivy

tf.math.xdivy(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: half, float32, float64, complex64, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.xlogy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/xlogy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/xlogy#aliases)

Returns 0 if x == 0, and x \* log(y) otherwise, elementwise.

### Aliases:

* tf.compat.v1.math.xlogy
* tf.compat.v2.math.xlogy
* tf.math.xlogy

tf.math.xlogy(  
    x,  
    y,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

#### Args:

* **x**: A Tensor. Must be one of the following types: half, float32, float64, complex64, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.zero\_fraction

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/zero_fraction#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/zero_fraction#aliases)

Returns the fraction of zeros in value.

### Aliases:

* tf.compat.v1.math.zero\_fraction
* tf.compat.v1.nn.zero\_fraction
* tf.compat.v2.math.zero\_fraction
* tf.compat.v2.nn.zero\_fraction
* tf.math.zero\_fraction
* tf.nn.zero\_fraction

tf.math.zero\_fraction(  
    value,  
    name=None  
)

Defined in [python/ops/nn\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/nn_impl.py).

If value is empty, the result is nan.

This is useful in summaries to measure and report sparsity. For example,

    z = tf.nn.relu(...)  
    summ = tf.compat.v1.summary.scalar('sparsity', tf.nn.zero\_fraction(z))

#### Args:

* **value**: A tensor of numeric type.
* **name**: A name for the operation (optional).

#### Returns:

The fraction of zeros in value, with type float32.

# tf.math.zeta

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/zeta#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/zeta#aliases)

Compute the Hurwitz zeta function ζ(x,q).

### Aliases:

* tf.compat.v1.math.zeta
* tf.compat.v1.zeta
* tf.compat.v2.math.zeta
* tf.math.zeta

tf.math.zeta(  
    x,  
    q,  
    name=None  
)

Defined in generated file: python/ops/gen\_math\_ops.py.

The Hurwitz zeta function is defined as:

ζ(x,q)=∑n=0∞(q+n)−x

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64.
* **q**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.