tf.add

```
add(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/gen_math_ops.py`.

See the guide: Math > Arithmetic Operators

Returns x + y element-wise.

NOTE: `Add` supports broadcasting. `AddN` does not. More about broadcasting here

Args:

- **`x`**: A `Tensor`. Must be one of the following types: `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.subtract

```
subtract(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/math_ops.py`.

See the guide: Math > Arithmetic Operators

Returns x - y element-wise.

NOTE: `tf.subtract` supports broadcasting. More about broadcasting here

Args:

- **`x`**: A `Tensor`. Must be one of the following types: `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.multiply

```
multiply(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/math_ops.py`.

See the guide: Math > Arithmetic Operators

Returns x * y element-wise.

NOTE: `tf.multiply` supports broadcasting. More about broadcasting here

Args:

- **`x`**: A `Tensor`. Must be one of the following types: `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.scalar_mul

```
scalar_mul(
    scalar,
    x
)
```

Defined in `tensorflow/python/ops/math_ops.py`.

See the guide: Math > Arithmetic Operators

Multiplies a scalar times a `Tensor` or `IndexedSlices` object.

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.

Returns:

`scalar * x` of the same type (`Tensor` or `IndexedSlices`) as `x`.

Raises:

- **`ValueError`**: if scalar is not a 0-D `scalar`.

tf.div

```
div(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/math_ops.py`.

See the guide: Math > Arithmetic Operators

Divides x / y elementwise (using Python 2 division operator semantics).

NOTE: Prefer using the Tensor division operator or tf.divide which obey Python division operator semantics.

This function divides `x` and `y`, forcing Python 2.7 semantics. That is, if one of `x` or `y` is a float, then the result will be a float. Otherwise, the output will be an integer type. Flooring semantics are used for integer division.

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` returns the quotient of x and y.

tf.divide

```
divide(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/math_ops.py`.

See the guide: Math > Arithmetic Operators

Computes Python style division of `x` by `y`.

tf.truediv

```
truediv(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/math_ops.py`.

See the guide: Math > Arithmetic Operators

Divides x / y elementwise (using Python 3 division operator semantics).

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 with `from __future__ import`

`division`. If you want integer division that rounds down, use `x // y` or `tf.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.floordiv

```
floordiv(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/math_ops.py`.

See the guide: Math > Arithmetic Operators

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

The same as `tf.div(x,y)` for integers, but uses `tf.floor(tf.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`.

Note that for efficiency, `floordiv` uses C semantics for negative numbers (unlike Python and Numpy).

`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 (except possibly towards zero for negative integers).

Raises:

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

tf.realdiv

```
realdiv(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/gen_math_ops.py`.

See the guide: Math > Arithmetic Operators

Returns x / y element-wise for real types.

If `x` and `y` are reals, this will return the floating-point division.

NOTE: `Div` supports broadcasting. More about broadcasting here

Args:

- **`x`**: A `Tensor`. Must be one of the following types: `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.truncatediv

```
truncatediv(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/gen_math_ops.py`.

See the guide: Math > Arithmetic Operators

Returns x / y element-wise for integer types.

Truncation designates that negative numbers will round fractional quantities toward zero. I.e. -7 / 5 = 1. This matches C semantics but it is different than Python semantics. See `FloorDiv` for a division function that matches Python Semantics.

NOTE: `TruncateDiv` supports broadcasting. More about broadcasting here

Args:

- **`x`**: A `Tensor`. Must be one of the following types: `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.floor_div

```
floor_div(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/gen_math_ops.py`.

See the guide: Math > Arithmetic Operators

Returns x // y element-wise.

NOTE: `FloorDiv` supports broadcasting. More about broadcasting here

Args:

- **`x`**: A `Tensor`. Must be one of the following types: `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.truncatemod

```
truncatemod(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/gen_math_ops.py`.

See the guide: Math > Arithmetic Operators

Returns element-wise remainder of division. This emulates C semantics in that

the result here is consistent with a truncating divide. E.g. `truncate(x / y) * y + truncate_mod(x, y) = x`.

NOTE: `TruncateMod` supports broadcasting. More about broadcasting here

Args:

- **`x`**: A `Tensor`. Must be one of the following types: `int32`, `int64`, `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.floormod

Aliases:

- `tf.floormod`
- `tf.mod`

```
floormod(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/gen_math_ops.py`.

See the guide: Math > Arithmetic Operators

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

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: `FloorMod` supports broadcasting. More about broadcasting here

Args:

- **`x`**: A `Tensor`. Must be one of the following types: `int32`, `int64`, `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.floormod

Aliases:

- `tf.floormod`
- `tf.mod`

```
floormod(
    x,
    y,
    name=None
)
```

Defined in `tensorflow/python/ops/gen_math_ops.py`.

See the guide: Math > Arithmetic Operators

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

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: `FloorMod` supports broadcasting. More about broadcasting here

Args:

- **`x`**: A `Tensor`. Must be one of the following types: `int32`, `int64`, `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.cross

```
cross(
    a,
    b,
    name=None
)
```

Defined in `tensorflow/python/ops/gen_math_ops.py`.

See the guide: Math > Arithmetic Operators

Compute the pairwise cross product.

`a` and `b` must be the same shape; they can either be simple 3-element vectors, or any shape where the innermost dimension is 3. In the latter case, each pair of corresponding 3-element vectors is cross-multiplied independently.

Args:

- **`a`**: A `Tensor`. Must be one of the following types: `float32`, `float64`, `int32`, `int64`, `uint8`, `int16`, `int8`, `uint16`, `half`. A tensor containing 3-element vectors.
- **`b`**: A `Tensor`. Must have the same type as `a`. Another tensor, of same type and shape as `a`.
- **`name`**: A name for the operation (optional).

Returns:

A `Tensor`. Has the same type as `a`. Pairwise cross product of the vectors in `a` and `b`.