tf.Dimension

Contents

Class **Dimension**

Defined in `tensorflow/python/framework/tensor_shape.py`.

See the guide: Building Graphs > Defining new operations

Represents the value of one dimension in a TensorShape.

Properties

`value`

The value of this dimension, or None if it is unknown.

Methods

`__init__`

```
__init__(value)
```

Creates a new Dimension with the given value.

`__add__`

```
__add__(other)
```

Returns the sum of `self` and `other`.

Dimensions are summed as follows:

```
tf.Dimension(m)    + tf.Dimension(n)    == tf.Dimension(m + n)
tf.Dimension(m)    + tf.Dimension(None) == tf.Dimension(None)
tf.Dimension(None) + tf.Dimension(n)    == tf.Dimension(None)
tf.Dimension(None) + tf.Dimension(None) == tf.Dimension(None)
```

Args:

- `other`: Another Dimension.

Returns:

A Dimension whose value is the sum of `self` and `other`.

`__div__`

```
__div__(other)
```

DEPRECATED: Use `__floordiv__` via `x // y` instead.

This function exists only for backwards compatibility purposes; new code should use `__floordiv__` via the syntax `x // y`. Using `x // y` communicates clearly that the result rounds down, and is forward compatible to Python 3.

Args:

- `other`: Another `Dimension`.

Returns:

A `Dimension` whose value is the integer quotient of `self` and `other`.

`__eq__`

```
__eq__(other)
```

Returns true if `other` has the same known value as this Dimension.

`__floordiv__`

```
__floordiv__(other)
```

Returns the quotient of `self` and `other` rounded down.

Dimensions are divided as follows:

```
tf.Dimension(m)    // tf.Dimension(n)    == tf.Dimension(m // n)
tf.Dimension(m)    // tf.Dimension(None) == tf.Dimension(None)
tf.Dimension(None) // tf.Dimension(n)    == tf.Dimension(None)
tf.Dimension(None) // tf.Dimension(None) == tf.Dimension(None)
```

Args:

- `other`: Another `Dimension`.

Returns:

A `Dimension` whose value is the integer quotient of `self` and `other`.

`__ge__`

```
__ge__(other)
```

Returns True if `self` is known to be greater than or equal to `other`.

Dimensions are compared as follows:

```
(tf.Dimension(m)    >= tf.Dimension(n))    == (m >= n)
(tf.Dimension(m)    >= tf.Dimension(None)) == None
(tf.Dimension(None) >= tf.Dimension(n))    == None
(tf.Dimension(None) >= tf.Dimension(None)) == None
```

Args:

- `other`: Another Dimension.

Returns:

The value of `self.value >= other.value` if both are known, otherwise None.

`__gt__`

```
__gt__(other)
```

Returns True if `self` is known to be greater than `other`.

Dimensions are compared as follows:

```
(tf.Dimension(m)    > tf.Dimension(n))    == (m > n)
(tf.Dimension(m)    > tf.Dimension(None)) == None
(tf.Dimension(None) > tf.Dimension(n))    == None
(tf.Dimension(None) > tf.Dimension(None)) == None
```

Args:

- `other`: Another Dimension.

Returns:

The value of `self.value > other.value` if both are known, otherwise None.

`__index__`

```
__index__()
```

`__int__`

```
__int__()
```

`__le__`

```
__le__(other)
```

Returns True if `self` is known to be less than or equal to `other`.

Dimensions are compared as follows:

```
(tf.Dimension(m)    <= tf.Dimension(n))    == (m <= n)
(tf.Dimension(m)    <= tf.Dimension(None)) == None
(tf.Dimension(None) <= tf.Dimension(n))    == None
(tf.Dimension(None) <= tf.Dimension(None)) == None
```

Args:

- `other`: Another Dimension.

Returns:

The value of `self.value <= other.value` if both are known, otherwise None.

__long__

```
__long__()
```

__lt__

```
__lt__(other)
```

Returns True if `self` is known to be less than `other`.

Dimensions are compared as follows:

```
(tf.Dimension(m)    < tf.Dimension(n))    == (m < n)
(tf.Dimension(m)    < tf.Dimension(None)) == None
(tf.Dimension(None) < tf.Dimension(n))    == None
(tf.Dimension(None) < tf.Dimension(None)) == None
```

Args:

- `other`: Another Dimension.

Returns:

The value of `self.value < other.value` if both are known, otherwise None.

__mod__

```
__mod__(other)
```

Returns `self` modulo `other.

Dimension moduli are computed as follows:

```
tf.Dimension(m)    % tf.Dimension(n)    == tf.Dimension(m % n)
tf.Dimension(m)    % tf.Dimension(None) == tf.Dimension(None)
tf.Dimension(None) % tf.Dimension(n)    == tf.Dimension(None)
tf.Dimension(None) % tf.Dimension(None) == tf.Dimension(None)
```

Args:

- `other`: Another Dimension.

Returns:

A Dimension whose value is `self` modulo `other`.

`__mul__`

```
__mul__(other)
```

Returns the product of `self` and `other`.

Dimensions are summed as follows:

```
tf.Dimension(m)    * tf.Dimension(n)    == tf.Dimension(m * n)
tf.Dimension(m)    * tf.Dimension(None) == tf.Dimension(None)
tf.Dimension(None) * tf.Dimension(n)    == tf.Dimension(None)
tf.Dimension(None) * tf.Dimension(None) == tf.Dimension(None)
```

Args:

- `other`: Another Dimension.

Returns:

A Dimension whose value is the product of `self` and `other`.

`__ne__`

```
__ne__(other)
```

Returns true if `other` has a different known value from `self`.

`__sub__`

```
__sub__(other)
```

Returns the subtraction of `other` from `self`.

Dimensions are subtracted as follows:

```
tf.Dimension(m)    - tf.Dimension(n)    == tf.Dimension(m - n)
tf.Dimension(m)    - tf.Dimension(None) == tf.Dimension(None)
tf.Dimension(None) - tf.Dimension(n)    == tf.Dimension(None)
tf.Dimension(None) - tf.Dimension(None) == tf.Dimension(None)
```

Args:

- `other`: Another Dimension.

Returns:

A Dimension whose value is the subtraction of sum of **`other`** from **`self`**.

assert_is_compatible_with

```
assert_is_compatible_with(other)
```

Raises an exception if **`other`** is not compatible with this Dimension.

Args:

- `other`: Another Dimension.

Raises:

- `ValueError`: If **`self`** and **`other`** are not compatible (see is_compatible_with).

is_compatible_with

```
is_compatible_with(other)
```

Returns true if **`other`** is compatible with this Dimension.

Two known Dimensions are compatible if they have the same value. An unknown Dimension is compatible with all other Dimensions.

Args:

- `other`: Another Dimension.

Returns:

True if this Dimension and **`other`** are compatible.

merge_with

```
merge_with(other)
```

Returns a Dimension that combines the information in **`self`** and **`other`**.

Dimensions are combined as follows:

```
tf.Dimension(n)   .merge_with(tf.Dimension(n))    == tf.Dimension(n)
tf.Dimension(n)   .merge_with(tf.Dimension(None)) == tf.Dimension(n)
tf.Dimension(None).merge_with(tf.Dimension(n))    == tf.Dimension(n)
tf.Dimension(None).merge_with(tf.Dimension(None)) == tf.Dimension(None)
tf.Dimension(n)   .merge_with(tf.Dimension(m))  # raises ValueError for n != m
```

Args:

- `other`: Another Dimension.

Returns:

A Dimension containing the combined information of `self` and `other`.

Raises:

- `ValueError`: If `self` and `other` are not compatible (see is_compatible_with).
