Module: tf.compat.v1.strings / tf.strings

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/strings#functions)
* Functions

Operations for working with string Tensors.

Functions

[as\_string(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/as_string): Converts each entry in the given tensor to strings. Supports many numeric

[bytes\_split(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/bytes_split): Split string elements of input into bytes.

[format(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/format): Formats a string template using a list of tensors.

[join(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/join): Joins the strings in the given list of string tensors into one tensor;

[length(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/strings/length): String lengths of input.

[lower(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/lower): TODO: add doc.

[reduce\_join(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/reduce_join): Joins a string Tensor across the given dimensions.

[regex\_full\_match(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/regex_full_match): Check if the input matches the regex pattern.

[regex\_replace(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/regex_replace): Replace elements of input matching regex pattern with rewrite.

[split(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/strings/split): Split elements of input based on sep.

[strip(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/strip): Strip leading and trailing whitespaces from the Tensor.

[substr(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/strings/substr): Return substrings from Tensor of strings.

[to\_hash\_bucket(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/string_to_hash_bucket): Converts each string in the input Tensor to its hash mod by a number of buckets.

[to\_hash\_bucket\_fast(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_hash_bucket_fast): Converts each string in the input Tensor to its hash mod by a number of buckets.

[to\_hash\_bucket\_strong(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_hash_bucket_strong): Converts each string in the input Tensor to its hash mod by a number of buckets.

[to\_number(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/string_to_number): Converts each string in the input Tensor to the specified numeric type.

[unicode\_decode(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_decode): Decodes each string in input into a sequence of Unicode code points.

[unicode\_decode\_with\_offsets(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_decode_with_offsets): Decodes each string into a sequence of code points with start offsets.

[unicode\_encode(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_encode): Encodes each sequence of Unicode code points in input into a string.

[unicode\_script(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_script): Determine the script codes of a given tensor of Unicode integer code points.

[unicode\_split(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_split): Splits each string in input into a sequence of Unicode code points.

[unicode\_split\_with\_offsets(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_split_with_offsets): Splits each string into a sequence of code points with start offsets.

[unicode\_transcode(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_transcode): Transcode the input text from a source encoding to a destination encoding.

[upper(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/upper): TODO: add doc.

# tf.compat.v1.strings.length

String lengths of input.

tf.compat.v1.strings.length(  
    input,  
    name=None,  
    unit='BYTE'  
)

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

Computes the length of each string given in the input tensor.

#### Args:

* **input**: A Tensor of type string. The string for which to compute the length.
* **unit**: An optional string from: "BYTE", "UTF8\_CHAR". Defaults to "BYTE". The unit that is counted to compute string length. One of: "BYTE" (for the number of bytes in each string) or "UTF8\_CHAR" (for the number of UTF-8 encoded Unicode code points in each string). Results are undefined if unit=UTF8\_CHAR and the input strings do not contain structurally valid UTF-8.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type int32.

# tf.compat.v1.strings.split

Split elements of input based on sep.

tf.compat.v1.strings.split(  
    input=None,  
    sep=None,  
    maxsplit=-1,  
    result\_type='SparseTensor',  
    source=None,  
    name=None  
)

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

Let N be the size of input (typically N will be the batch size). Split each element of input based on sep and return a SparseTensor or RaggedTensor containing the split tokens. Empty tokens are ignored.

#### Examples:

>>> tf.strings.split(['hello world', 'a b c'])  
tf.SparseTensor(indices=[[0, 0], [0, 1], [1, 0], [1, 1], [1, 2]],  
                values=['hello', 'world', 'a', 'b', 'c']  
                dense\_shape=[2, 3])  
  
>>> tf.strings.split(['hello world', 'a b c'], result\_type="RaggedTensor")  
<tf.RaggedTensor [['hello', 'world'], ['a', 'b', 'c']]>

If sep is given, consecutive delimiters are not grouped together and are deemed to delimit empty strings. For example, input of "1<>2<><>3" and sep of "<>" returns ["1", "2", "", "3"]. If sep is None or an empty string, consecutive whitespace are regarded as a single separator, and the result will contain no empty strings at the start or end if the string has leading or trailing whitespace.

Note that the above mentioned behavior matches python's str.split.

#### Args:

* **input**: A string Tensor of rank N, the strings to split. If rank(input) is not known statically, then it is assumed to be 1.
* **sep**: 0-D string Tensor, the delimiter character.
* **maxsplit**: An int. If maxsplit > 0, limit of the split of the result.
* **result\_type**: The tensor type for the result: one of "RaggedTensor" or "SparseTensor".
* **source**: alias for "input" argument.
* **name**: A name for the operation (optional).

#### Raises:

* **ValueError**: If sep is not a string.

#### Returns:

A SparseTensor or RaggedTensor of rank N+1, the strings split according to the delimiter.

# tf.compat.v1.strings.substr

Return substrings from Tensor of strings.

tf.compat.v1.strings.substr(  
    input,  
    pos,  
    len,  
    name=None,  
    unit='BYTE'  
)

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

For each string in the input Tensor, creates a substring starting at index pos with a total length of len.

If len defines a substring that would extend beyond the length of the input string, then as many characters as possible are used.

A negative pos indicates distance within the string backwards from the end.

If pos specifies an index which is out of range for any of the input strings, then an InvalidArgumentError is thrown.

pos and len must have the same shape, otherwise a ValueError is thrown on Op creation.

NOTE: Substr supports broadcasting up to two dimensions. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

Examples

Using scalar pos and len:

input = [b'Hello', b'World']  
position = 1  
length = 3  
  
output = [b'ell', b'orl']

Using pos and len with same shape as input:

input = [[b'ten', b'eleven', b'twelve'],  
         [b'thirteen', b'fourteen', b'fifteen'],  
         [b'sixteen', b'seventeen', b'eighteen']]  
position = [[1, 2, 3],  
            [1, 2, 3],  
            [1, 2, 3]]  
length =   [[2, 3, 4],  
            [4, 3, 2],  
            [5, 5, 5]]  
  
output = [[b'en', b'eve', b'lve'],  
          [b'hirt', b'urt', b'te'],  
          [b'ixtee', b'vente', b'hteen']]

Broadcasting pos and len onto input:

input = [[b'ten', b'eleven', b'twelve'],  
         [b'thirteen', b'fourteen', b'fifteen'],  
         [b'sixteen', b'seventeen', b'eighteen'],  
         [b'nineteen', b'twenty', b'twentyone']]  
position = [1, 2, 3]  
length =   [1, 2, 3]  
  
output = [[b'e', b'ev', b'lve'],  
          [b'h', b'ur', b'tee'],  
          [b'i', b've', b'hte'],  
          [b'i', b'en', b'nty']]

Broadcasting input onto pos and len:

input = b'thirteen'  
position = [1, 5, 7]  
length =   [3, 2, 1]  
  
output = [b'hir', b'ee', b'n']

#### Args:

* **input**: A Tensor of type string. Tensor of strings
* **pos**: A Tensor. Must be one of the following types: int32, int64. Scalar defining the position of first character in each substring
* **len**: A Tensor. Must have the same type as pos. Scalar defining the number of characters to include in each substring
* **unit**: An optional string from: "BYTE", "UTF8\_CHAR". Defaults to "BYTE". The unit that is used to create the substring. One of: "BYTE" (for defining position and length by bytes) or "UTF8\_CHAR" (for the UTF-8 encoded Unicode code points). The default is "BYTE". Results are undefined if unit=UTF8\_CHAR and the input strings do not contain structurally valid UTF-8.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.as\_string

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

Converts each entry in the given tensor to strings. Supports many numeric

### Aliases:

* tf.as\_string
* tf.compat.v1.as\_string
* tf.compat.v1.dtypes.as\_string
* tf.compat.v1.strings.as\_string
* tf.compat.v2.as\_string
* tf.compat.v2.strings.as\_string
* tf.strings.as\_string

tf.strings.as\_string(  
    input,  
    precision=-1,  
    scientific=False,  
    shortest=False,  
    width=-1,  
    fill='',  
    name=None  
)

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

types and boolean.

#### Args:

* **input**: A Tensor. Must be one of the following types: int8, int16, int32, int64, complex64, complex128, float32, float64, bool.
* **precision**: An optional int. Defaults to -1. The post-decimal precision to use for floating point numbers. Only used if precision > -1.
* **scientific**: An optional bool. Defaults to False. Use scientific notation for floating point numbers.
* **shortest**: An optional bool. Defaults to False. Use shortest representation (either scientific or standard) for floating point numbers.
* **width**: An optional int. Defaults to -1. Pad pre-decimal numbers to this width. Applies to both floating point and integer numbers. Only used if width > -1.
* **fill**: An optional string. Defaults to "". The value to pad if width > -1. If empty, pads with spaces. Another typical value is '0'. String cannot be longer than 1 character.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.bytes\_split

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

Split string elements of input into bytes.

### Aliases:

* tf.compat.v1.strings.bytes\_split
* tf.compat.v2.strings.bytes\_split
* tf.strings.bytes\_split

tf.strings.bytes\_split(  
    input,  
    name=None  
)

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

#### Examples:

>>> tf.strings.to\_bytes('hello')  
['h', 'e', 'l', 'l', 'o']  
>>> tf.strings.to\_bytes(['hello', '123'])  
<RaggedTensor [['h', 'e', 'l', 'l', 'o'], ['1', '2', '3']]>

Note that this op splits strings into bytes, not unicode characters. To split strings into unicode characters, use [tf.strings.unicode\_split](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_split).

See also: [tf.io.decode\_raw](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_raw), [tf.strings.split](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/split), [tf.strings.unicode\_split](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_split).

#### Args:

* **input**: A string Tensor or RaggedTensor: the strings to split. Must have a statically known rank (N).
* **name**: A name for the operation (optional).

#### Returns:

A RaggedTensor of rank N+1: the bytes that make up the soruce strings.

# tf.strings.format

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

Formats a string template using a list of tensors.

### Aliases:

* tf.compat.v1.strings.format
* tf.compat.v2.strings.format
* tf.strings.format

tf.strings.format(  
    template,  
    inputs,  
    placeholder='{}',  
    summarize=3,  
    name=None  
)

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

Formats a string template using a list of tensors, abbreviating tensors by only printing the first and last summarize elements of each dimension (recursively). If formatting only one tensor into a template, the tensor does not have to be wrapped in a list.

#### Example:

Formatting a single-tensor template:

sess = tf.compat.v1.Session()  
with sess.as\_default():  
    tensor = tf.range(10)  
    formatted = tf.strings.format("tensor: {}, suffix", tensor)  
    out = sess.run(formatted)  
    expected = "tensor: [0 1 2 ... 7 8 9], suffix"  
  
    assert(out.decode() == expected)

Formatting a multi-tensor template:

sess = tf.compat.v1.Session()  
with sess.as\_default():  
    tensor\_one = tf.reshape(tf.range(100), [10, 10])  
    tensor\_two = tf.range(10)  
    formatted = tf.strings.format("first: {}, second: {}, suffix",  
      (tensor\_one, tensor\_two))  
  
    out = sess.run(formatted)  
    expected = ("first: [[0 1 2 ... 7 8 9]\n"  
          " [10 11 12 ... 17 18 19]\n"  
          " [20 21 22 ... 27 28 29]\n"  
          " ...\n"  
          " [70 71 72 ... 77 78 79]\n"  
          " [80 81 82 ... 87 88 89]\n"  
          " [90 91 92 ... 97 98 99]], second: [0 1 2 ... 7 8 9], suffix")  
  
    assert(out.decode() == expected)

#### Args:

* **template**: A string template to format tensor values into.
* **inputs**: A list of Tensor objects, or a single Tensor. The list of tensors to format into the template string. If a solitary tensor is passed in, the input tensor will automatically be wrapped as a list.
* **placeholder**: An optional string. Defaults to {}. At each placeholder occurring in the template, a subsequent tensor will be inserted.
* **summarize**: An optional int. Defaults to 3. When formatting the tensors, show the first and last summarize entries of each tensor dimension (recursively). If set to -1, all elements of the tensor will be shown.
* **name**: A name for the operation (optional).

#### Returns:

A scalar Tensor of type string.

#### Raises:

* **ValueError**: if the number of placeholders does not match the number of inputs.

# tf.strings.join

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

Joins the strings in the given list of string tensors into one tensor;

### Aliases:

* tf.compat.v1.string\_join
* tf.compat.v1.strings.join
* tf.compat.v2.strings.join
* tf.strings.join

tf.strings.join(  
    inputs,  
    separator='',  
    name=None  
)

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

### Used in the guide:

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

with the given separator (default is an empty separator).

#### Args:

* **inputs**: A list of at least 1 Tensor objects with type string. A list of string tensors. The tensors must all have the same shape, or be scalars. Scalars may be mixed in; these will be broadcast to the shape of non-scalar inputs.
* **separator**: An optional string. Defaults to "". string, an optional join separator.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.length

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

### Aliases:

* tf.compat.v2.strings.length
* tf.strings.length

tf.strings.length(  
    input,  
    unit='BYTE',  
    name=None  
)

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

### Used in the tutorials:

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

# tf.strings.lower

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

TODO: add doc.

### Aliases:

* tf.compat.v1.strings.lower
* tf.compat.v2.strings.lower
* tf.strings.lower

tf.strings.lower(  
    input,  
    encoding='',  
    name=None  
)

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

#### Args:

* **input**: A Tensor of type string.
* **encoding**: An optional string. Defaults to "".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.reduce\_join

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

### Aliases:

* tf.compat.v2.strings.reduce\_join
* tf.strings.reduce\_join

tf.strings.reduce\_join(  
    inputs,  
    axis=None,  
    keepdims=False,  
    separator='',  
    name=None  
)

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

# tf.strings.regex\_full\_match

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

Check if the input matches the regex pattern.

### Aliases:

* tf.compat.v1.strings.regex\_full\_match
* tf.compat.v2.strings.regex\_full\_match
* tf.strings.regex\_full\_match

tf.strings.regex\_full\_match(  
    input,  
    pattern,  
    name=None  
)

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

The input is a string tensor of any shape. The pattern is a scalar string tensor which is applied to every element of the input tensor. The boolean values (True or False) of the output tensor indicate if the input matches the regex pattern provided.

The pattern follows the re2 syntax (https://github.com/google/re2/wiki/Syntax)

#### Args:

* **input**: A Tensor of type string. A string tensor of the text to be processed.
* **pattern**: A Tensor of type string. A scalar string tensor containing the regular expression to match the input.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.strings.regex\_replace

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

Replace elements of input matching regex pattern with rewrite.

### Aliases:

* tf.compat.v1.regex\_replace
* tf.compat.v1.strings.regex\_replace
* tf.compat.v2.strings.regex\_replace
* tf.strings.regex\_replace

tf.strings.regex\_replace(  
    input,  
    pattern,  
    rewrite,  
    replace\_global=True,  
    name=None  
)

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

### Used in the tutorials:

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

#### Args:

* **input**: string Tensor, the source strings to process.
* **pattern**: string or scalar string Tensor, regular expression to use, see more details at https://github.com/google/re2/wiki/Syntax
* **rewrite**: string or scalar string Tensor, value to use in match replacement, supports backslash-escaped digits (\1 to \9) can be to insert text matching corresponding parenthesized group.
* **replace\_global**: bool, if True replace all non-overlapping matches, else replace only the first match.
* **name**: A name for the operation (optional).

#### Returns:

string Tensor of the same shape as input with specified replacements.

# tf.strings.split

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

Split elements of input based on sep into a RaggedTensor.

### Aliases:

* tf.compat.v2.strings.split
* tf.strings.split

tf.strings.split(  
    input,  
    sep=None,  
    maxsplit=-1,  
    name=None  
)

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

Let N be the size of input (typically N will be the batch size). Split each element of input based on sep and return a SparseTensor or RaggedTensor containing the split tokens. Empty tokens are ignored.

#### Example:

>>> tf.strings.split('hello world')  
<Tensor ['hello', 'world']>  
>>> tf.strings.split(['hello world', 'a b c'])  
<tf.RaggedTensor [['hello', 'world'], ['a', 'b', 'c']]>

If sep is given, consecutive delimiters are not grouped together and are deemed to delimit empty strings. For example, input of "1<>2<><>3" and sep of "<>" returns ["1", "2", "", "3"]. If sep is None or an empty string, consecutive whitespace are regarded as a single separator, and the result will contain no empty strings at the start or end if the string has leading or trailing whitespace.

Note that the above mentioned behavior matches python's str.split.

#### Args:

* **input**: A string Tensor of rank N, the strings to split. If rank(input) is not known statically, then it is assumed to be 1.
* **sep**: 0-D string Tensor, the delimiter string.
* **maxsplit**: An int. If maxsplit > 0, limit of the split of the result.
* **name**: A name for the operation (optional).

#### Raises:

* **ValueError**: If sep is not a string.

#### Returns:

A RaggedTensor of rank N+1, the strings split according to the delimiter.

# tf.strings.strip

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

Strip leading and trailing whitespaces from the Tensor.

### Aliases:

* tf.compat.v1.string\_strip
* tf.compat.v1.strings.strip
* tf.compat.v2.strings.strip
* tf.strings.strip

tf.strings.strip(  
    input,  
    name=None  
)

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

#### Args:

* **input**: A Tensor of type string. A string Tensor of any shape.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.substr

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

Return substrings from Tensor of strings.

### Aliases:

* tf.compat.v2.strings.substr
* tf.strings.substr

tf.strings.substr(  
    input,  
    pos,  
    len,  
    unit='BYTE',  
    name=None  
)

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

### Used in the guide:

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

### Used in the tutorials:

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

For each string in the input Tensor, creates a substring starting at index pos with a total length of len.

If len defines a substring that would extend beyond the length of the input string, then as many characters as possible are used.

A negative pos indicates distance within the string backwards from the end.

If pos specifies an index which is out of range for any of the input strings, then an InvalidArgumentError is thrown.

pos and len must have the same shape, otherwise a ValueError is thrown on Op creation.

NOTE: Substr supports broadcasting up to two dimensions. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

Examples

Using scalar pos and len:

input = [b'Hello', b'World']  
position = 1  
length = 3  
  
output = [b'ell', b'orl']

Using pos and len with same shape as input:

input = [[b'ten', b'eleven', b'twelve'],  
         [b'thirteen', b'fourteen', b'fifteen'],  
         [b'sixteen', b'seventeen', b'eighteen']]  
position = [[1, 2, 3],  
            [1, 2, 3],  
            [1, 2, 3]]  
length =   [[2, 3, 4],  
            [4, 3, 2],  
            [5, 5, 5]]  
  
output = [[b'en', b'eve', b'lve'],  
          [b'hirt', b'urt', b'te'],  
          [b'ixtee', b'vente', b'hteen']]

Broadcasting pos and len onto input:

input = [[b'ten', b'eleven', b'twelve'],  
         [b'thirteen', b'fourteen', b'fifteen'],  
         [b'sixteen', b'seventeen', b'eighteen'],  
         [b'nineteen', b'twenty', b'twentyone']]  
position = [1, 2, 3]  
length =   [1, 2, 3]  
  
output = [[b'e', b'ev', b'lve'],  
          [b'h', b'ur', b'tee'],  
          [b'i', b've', b'hte'],  
          [b'i', b'en', b'nty']]

Broadcasting input onto pos and len:

input = b'thirteen'  
position = [1, 5, 7]  
length =   [3, 2, 1]  
  
output = [b'hir', b'ee', b'n']

#### Args:

* **input**: A Tensor of type string. Tensor of strings
* **pos**: A Tensor. Must be one of the following types: int32, int64. Scalar defining the position of first character in each substring
* **len**: A Tensor. Must have the same type as pos. Scalar defining the number of characters to include in each substring
* **unit**: An optional string from: "BYTE", "UTF8\_CHAR". Defaults to "BYTE". The unit that is used to create the substring. One of: "BYTE" (for defining position and length by bytes) or "UTF8\_CHAR" (for the UTF-8 encoded Unicode code points). The default is "BYTE". Results are undefined if unit=UTF8\_CHAR and the input strings do not contain structurally valid UTF-8.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.to\_hash\_bucket

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

Converts each string in the input Tensor to its hash mod by a number of buckets.

### Aliases:

* tf.compat.v2.strings.to\_hash\_bucket
* tf.strings.to\_hash\_bucket

tf.strings.to\_hash\_bucket(  
    input,  
    num\_buckets,  
    name=None  
)

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

The hash function is deterministic on the content of the string within the process.

Note that the hash function may change from time to time. This functionality will be deprecated and it's recommended to use tf.strings.to\_hash\_bucket\_fast() or tf.strings.to\_hash\_bucket\_strong().

#### Args:

* **input**: A Tensor of type string.
* **num\_buckets**: An int that is >= 1. The number of buckets.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type int64.

# tf.strings.to\_hash\_bucket\_fast

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

Converts each string in the input Tensor to its hash mod by a number of buckets.

### Aliases:

* tf.compat.v1.string\_to\_hash\_bucket\_fast
* tf.compat.v1.strings.to\_hash\_bucket\_fast
* tf.compat.v2.strings.to\_hash\_bucket\_fast
* tf.strings.to\_hash\_bucket\_fast

tf.strings.to\_hash\_bucket\_fast(  
    input,  
    num\_buckets,  
    name=None  
)

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

### Used in the guide:

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

The hash function is deterministic on the content of the string within the process and will never change. However, it is not suitable for cryptography. This function may be used when CPU time is scarce and inputs are trusted or unimportant. There is a risk of adversaries constructing inputs that all hash to the same bucket. To prevent this problem, use a strong hash function withtf.string\_to\_hash\_bucket\_strong.

#### Args:

* **input**: A Tensor of type string. The strings to assign a hash bucket.
* **num\_buckets**: An int that is >= 1. The number of buckets.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type int64.

# tf.strings.to\_hash\_bucket\_strong

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

Converts each string in the input Tensor to its hash mod by a number of buckets.

### Aliases:

* tf.compat.v1.string\_to\_hash\_bucket\_strong
* tf.compat.v1.strings.to\_hash\_bucket\_strong
* tf.compat.v2.strings.to\_hash\_bucket\_strong
* tf.strings.to\_hash\_bucket\_strong

tf.strings.to\_hash\_bucket\_strong(  
    input,  
    num\_buckets,  
    key,  
    name=None  
)

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

The hash function is deterministic on the content of the string within the process. The hash function is a keyed hash function, where attribute key defines the key of the hash function. key is an array of 2 elements.

A strong hash is important when inputs may be malicious, e.g. URLs with additional components. Adversaries could try to make their inputs hash to the same bucket for a denial-of-service attack or to skew the results. A strong hash can be used to make it difficult to find inputs with a skewed hash value distribution over buckets. This requires that the hash function is seeded by a high-entropy (random) "key" unknown to the adversary.

The additional robustness comes at a cost of roughly 4x higher compute time than tf.string\_to\_hash\_bucket\_fast.

#### Args:

* **input**: A Tensor of type string. The strings to assign a hash bucket.
* **num\_buckets**: An int that is >= 1. The number of buckets.
* **key**: A list of ints. The key used to seed the hash function, passed as a list of two uint64 elements.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type int64.

# tf.strings.to\_number

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

Converts each string in the input Tensor to the specified numeric type.

### Aliases:

* tf.compat.v2.strings.to\_number
* tf.strings.to\_number

tf.strings.to\_number(  
    input,  
    out\_type=tf.dtypes.float32,  
    name=None  
)

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

### Used in the guide:

* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

(Note that int32 overflow results in an error while float overflow results in a rounded value.)

#### Args:

* **input**: A Tensor of type string.
* **out\_type**: An optional [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) from: tf.float32, tf.float64, tf.int32, tf.int64. Defaults to [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32). The numeric type to interpret each string in string\_tensor as.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type out\_type.

# tf.strings.unicode\_decode

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

Decodes each string in input into a sequence of Unicode code points.

### Aliases:

* tf.compat.v1.strings.unicode\_decode
* tf.compat.v2.strings.unicode\_decode
* tf.strings.unicode\_decode

tf.strings.unicode\_decode(  
    input,  
    input\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    replace\_control\_characters=False,  
    name=None  
)

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

### Used in the tutorials:

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

result[i1...iN, j] is the Unicode codepoint for the jth character in input[i1...iN], when decoded using input\_encoding.

#### Args:

* **input**: An N dimensional potentially ragged string tensor with shape [D1...DN]. N must be statically known.
* **input\_encoding**: String name for the unicode encoding that should be used to decode each string.
* **errors**: Specifies the response when an input string can't be converted using the indicated encoding. One of:
  + 'strict': Raise an exception for any illegal substrings.
  + 'replace': Replace illegal substrings with replacement\_char.
  + 'ignore': Skip illegal substrings.
* **replacement\_char**: The replacement codepoint to be used in place of invalid substrings in input when errors='replace'; and in place of C0 control characters in input when replace\_control\_characters=True.
* **replace\_control\_characters**: Whether to replace the C0 control characters (U+0000 - U+001F) with the replacement\_char.
* **name**: A name for the operation (optional).

#### Returns:

A N+1 dimensional int32 tensor with shape [D1...DN, (num\_chars)]. The returned tensor is a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) if input is a scalar, or a [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) otherwise.

#### Example:

>>> input = [s.encode('utf8') for s in (u'G\xf6\xf6dnight', u'\U0001f60a')]  
>>> tf.strings.unicode\_decode(input, 'UTF-8').tolist()  
[[71, 246, 246, 100, 110, 105, 103, 104, 116], [128522]]

# tf.strings.unicode\_decode\_with\_offsets

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

Decodes each string into a sequence of code points with start offsets.

### Aliases:

* tf.compat.v1.strings.unicode\_decode\_with\_offsets
* tf.compat.v2.strings.unicode\_decode\_with\_offsets
* tf.strings.unicode\_decode\_with\_offsets

tf.strings.unicode\_decode\_with\_offsets(  
    input,  
    input\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    replace\_control\_characters=False,  
    name=None  
)

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

### Used in the tutorials:

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

This op is similar to tf.strings.decode(...), but it also returns the start offset for each character in its respective string. This information can be used to align the characters with the original byte sequence.

Returns a tuple (codepoints, start\_offsets) where:

* codepoints[i1...iN, j] is the Unicode codepoint for the jth character in input[i1...iN], when decoded using input\_encoding.
* start\_offsets[i1...iN, j] is the start byte offset for the jth character in input[i1...iN], when decoded using input\_encoding.

#### Args:

* **input**: An N dimensional potentially ragged string tensor with shape [D1...DN]. N must be statically known.
* **input\_encoding**: String name for the unicode encoding that should be used to decode each string.
* **errors**: Specifies the response when an input string can't be converted using the indicated encoding. One of:
  + 'strict': Raise an exception for any illegal substrings.
  + 'replace': Replace illegal substrings with replacement\_char.
  + 'ignore': Skip illegal substrings.
* **replacement\_char**: The replacement codepoint to be used in place of invalid substrings in input when errors='replace'; and in place of C0 control characters in input when replace\_control\_characters=True.
* **replace\_control\_characters**: Whether to replace the C0 control characters (U+0000 - U+001F) with the replacement\_char.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of N+1 dimensional tensors (codepoints, start\_offsets).

* codepoints is an int32 tensor with shape [D1...DN, (num\_chars)].
* offsets is an int64 tensor with shape [D1...DN, (num\_chars)].

The returned tensors are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor)s if input is a scalar, or [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor)s otherwise.

#### Example:

>>> input = [s.encode('utf8') for s in (u'G\xf6\xf6dnight', u'\U0001f60a')]  
>>> result = tf.strings.unicode\_decode\_with\_offsets(input, 'UTF-8')  
>>> result[0].tolist()  # codepoints  
[[71, 246, 246, 100, 110, 105, 103, 104, 116], [128522]]  
>>> result[1].tolist()  # offsets  
[0, 1, 3, 5, 6, 7, 8, 9, 10], [0]]

# tf.strings.unicode\_encode

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

Encodes each sequence of Unicode code points in input into a string.

### Aliases:

* tf.compat.v1.strings.unicode\_encode
* tf.compat.v2.strings.unicode\_encode
* tf.strings.unicode\_encode

tf.strings.unicode\_encode(  
    input,  
    output\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    name=None  
)

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

### Used in the tutorials:

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

result[i1...iN] is the string formed by concatenating the Unicode codepoints input[1...iN, :], encoded using output\_encoding.

#### Args:

* **input**: An N+1 dimensional potentially ragged integer tensor with shape [D1...DN, num\_chars].
* **output\_encoding**: Unicode encoding that should be used to encode each codepoint sequence. Can be "UTF-8", "UTF-16-BE", or "UTF-32-BE".
* **errors**: Specifies the response when an invalid codepoint is encountered (optional). One of: \* 'replace': Replace invalid codepoint with the replacement\_char. (default) \* 'ignore': Skip invalid codepoints. \* 'strict': Raise an exception for any invalid codepoint.
* **replacement\_char**: The replacement character codepoint to be used in place of any invalid input when errors='replace'. Any valid unicode codepoint may be used. The default value is the default unicode replacement character which is 0xFFFD (U+65533).
* **name**: A name for the operation (optional).

#### Returns:

A N dimensional string tensor with shape [D1...DN].

#### Example:

  >>> input = [[71, 246, 246, 100, 110, 105, 103, 104, 116], [128522]]  
  >>> unicode\_encode(input, 'UTF-8')  
  ['G\xc3\xb6\xc3\xb6dnight', '\xf0\x9f\x98\x8a']

# tf.strings.unicode\_script

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

Determine the script codes of a given tensor of Unicode integer code points.

### Aliases:

* tf.compat.v1.strings.unicode\_script
* tf.compat.v2.strings.unicode\_script
* tf.strings.unicode\_script

tf.strings.unicode\_script(  
    input,  
    name=None  
)

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

### Used in the tutorials:

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

This operation converts Unicode code points to script codes corresponding to each code point. Script codes correspond to International Components for Unicode (ICU) UScriptCode values. See http://icu-project.org/apiref/icu4c/uscript\_8h.html. Returns -1 (USCRIPT\_INVALID\_CODE) for invalid codepoints. Output shape will match input shape.

#### Args:

* **input**: A Tensor of type int32. A Tensor of int32 Unicode code points.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type int32.

# tf.strings.unicode\_split

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

Splits each string in input into a sequence of Unicode code points.

### Aliases:

* tf.compat.v1.strings.unicode\_split
* tf.compat.v2.strings.unicode\_split
* tf.strings.unicode\_split

tf.strings.unicode\_split(  
    input,  
    input\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    name=None  
)

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

### Used in the tutorials:

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

result[i1...iN, j] is the substring of input[i1...iN] that encodes its jth character, when decoded using input\_encoding.

#### Args:

* **input**: An N dimensional potentially ragged string tensor with shape [D1...DN]. N must be statically known.
* **input\_encoding**: String name for the unicode encoding that should be used to decode each string.
* **errors**: Specifies the response when an input string can't be converted using the indicated encoding. One of:
  + 'strict': Raise an exception for any illegal substrings.
  + 'replace': Replace illegal substrings with replacement\_char.
  + 'ignore': Skip illegal substrings.
* **replacement\_char**: The replacement codepoint to be used in place of invalid substrings in input when errors='replace'.
* **name**: A name for the operation (optional).

#### Returns:

A N+1 dimensional int32 tensor with shape [D1...DN, (num\_chars)]. The returned tensor is a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) if input is a scalar, or a [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) otherwise.

#### Example:

>>> input = [s.encode('utf8') for s in (u'G\xf6\xf6dnight', u'\U0001f60a')]  
>>> tf.strings.unicode\_split(input, 'UTF-8').tolist()  
[['G', '\xc3\xb6', '\xc3\xb6', 'd', 'n', 'i', 'g', 'h', 't'],  
 ['\xf0\x9f\x98\x8a']]

# tf.strings.unicode\_split\_with\_offsets

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

Splits each string into a sequence of code points with start offsets.

### Aliases:

* tf.compat.v1.strings.unicode\_split\_with\_offsets
* tf.compat.v2.strings.unicode\_split\_with\_offsets
* tf.strings.unicode\_split\_with\_offsets

tf.strings.unicode\_split\_with\_offsets(  
    input,  
    input\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    name=None  
)

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

This op is similar to tf.strings.decode(...), but it also returns the start offset for each character in its respective string. This information can be used to align the characters with the original byte sequence.

Returns a tuple (chars, start\_offsets) where:

* chars[i1...iN, j] is the substring of input[i1...iN] that encodes its jth character, when decoded using input\_encoding.
* start\_offsets[i1...iN, j] is the start byte offset for the jth character in input[i1...iN], when decoded using input\_encoding.

#### Args:

* **input**: An N dimensional potentially ragged string tensor with shape [D1...DN]. N must be statically known.
* **input\_encoding**: String name for the unicode encoding that should be used to decode each string.
* **errors**: Specifies the response when an input string can't be converted using the indicated encoding. One of:
  + 'strict': Raise an exception for any illegal substrings.
  + 'replace': Replace illegal substrings with replacement\_char.
  + 'ignore': Skip illegal substrings.
* **replacement\_char**: The replacement codepoint to be used in place of invalid substrings in input when errors='replace'.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of N+1 dimensional tensors (codepoints, start\_offsets).

* codepoints is an int32 tensor with shape [D1...DN, (num\_chars)].
* offsets is an int64 tensor with shape [D1...DN, (num\_chars)].

The returned tensors are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor)s if input is a scalar, or [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor)s otherwise.

#### Example:

>>> input = [s.encode('utf8') for s in (u'G\xf6\xf6dnight', u'\U0001f60a')]  
>>> result = tf.strings.unicode\_split\_with\_offsets(input, 'UTF-8')  
>>> result[0].tolist()  # character substrings  
[['G', '\xc3\xb6', '\xc3\xb6', 'd', 'n', 'i', 'g', 'h', 't'],  
 ['\xf0\x9f\x98\x8a']]  
>>> result[1].tolist()  # offsets  
[0, 1, 3, 5, 6, 7, 8, 9, 10], [0]]

# tf.strings.unicode\_transcode

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

Transcode the input text from a source encoding to a destination encoding.

### Aliases:

* tf.compat.v1.strings.unicode\_transcode
* tf.compat.v2.strings.unicode\_transcode
* tf.strings.unicode\_transcode

tf.strings.unicode\_transcode(  
    input,  
    input\_encoding,  
    output\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    replace\_control\_characters=False,  
    name=None  
)

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

### Used in the tutorials:

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

The input is a string tensor of any shape. The output is a string tensor of the same shape containing the transcoded strings. Output strings are always valid unicode. If the input contains invalid encoding positions, the errors attribute sets the policy for how to deal with them. If the default error-handling policy is used, invalid formatting will be substituted in the output by the replacement\_char. If the errors policy is to ignore, any invalid encoding positions in the input are skipped and not included in the output. If it set to strict then any invalid formatting will result in an InvalidArgument error.

This operation can be used with output\_encoding = input\_encoding to enforce correct formatting for inputs even if they are already in the desired encoding.

If the input is prefixed by a Byte Order Mark needed to determine encoding (e.g. if the encoding is UTF-16 and the BOM indicates big-endian), then that BOM will be consumed and not emitted into the output. If the input encoding is marked with an explicit endianness (e.g. UTF-16-BE), then the BOM is interpreted as a non-breaking-space and is preserved in the output (including always for UTF-8).

The end result is that if the input is marked as an explicit endianness the transcoding is faithful to all codepoints in the source. If it is not marked with an explicit endianness, the BOM is not considered part of the string itself but as metadata, and so is not preserved in the output.

#### Args:

* **input**: A Tensor of type string. The text to be processed. Can have any shape.
* **input\_encoding**: A string. Text encoding of the input strings. This is any of the encodings supported by ICU ucnv algorithmic converters. Examples: "UTF-16", "US ASCII", "UTF-8".
* **output\_encoding**: A string from: "UTF-8", "UTF-16-BE", "UTF-32-BE". The unicode encoding to use in the output. Must be one of "UTF-8", "UTF-16-BE", "UTF-32-BE". Multi-byte encodings will be big-endian.
* **errors**: An optional string from: "strict", "replace", "ignore". Defaults to "replace". Error handling policy when there is invalid formatting found in the input. The value of 'strict' will cause the operation to produce a InvalidArgument error on any invalid input formatting. A value of 'replace' (the default) will cause the operation to replace any invalid formatting in the input with the replacement\_char codepoint. A value of 'ignore' will cause the operation to skip any invalid formatting in the input and produce no corresponding output character.
* **replacement\_char**: An optional int. Defaults to 65533. The replacement character codepoint to be used in place of any invalid formatting in the input when errors='replace'. Any valid unicode codepoint may be used. The default value is the default unicode replacement character is 0xFFFD or U+65533.)

Note that for UTF-8, passing a replacement character expressible in 1 byte, such as ' ', will preserve string alignment to the source since invalid bytes will be replaced with a 1-byte replacement. For UTF-16-BE and UTF-16-LE, any 1 or 2 byte replacement character will preserve byte alignment to the source.

* **replace\_control\_characters**: An optional bool. Defaults to False. Whether to replace the C0 control characters (00-1F) with the replacement\_char. Default is false.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.upper

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

TODO: add doc.

### Aliases:

* tf.compat.v1.strings.upper
* tf.compat.v2.strings.upper
* tf.strings.upper

tf.strings.upper(  
    input,  
    encoding='',  
    name=None  
)

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

#### Args:

* **input**: A Tensor of type string.
* **encoding**: An optional string. Defaults to "".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.