

[Discover Packages](#) > [Standard library](#) > [strings](#) 

strings


package

standard library

Version: [go1.20.1](#) **Latest** | Published: Feb 14, 2023 | License: [BSD-3-Clause](#) | Imports: 7 |

Imported by: 1,238,971

Details

 Valid [go.mod](#) file  Redistributable license  Tagged version  Stable version [Learn more](#)

Repository

cs.opensource.google/go/go

Links

 [Report a Vulnerability](#) Documentation 

<> Documentation

Overview

Package strings implements simple functions to manipulate UTF-8 encoded strings.

For information about UTF-8 strings in Go, see <https://blog.golang.org/strings>.

Index

[func Clone\(s string\) string](#)[func Compare\(a, b string\) int](#)[func Contains\(s, substr string\) bool](#)[func ContainsAny\(s, chars string\) bool](#)[func ContainsRune\(s string, r rune\) bool](#)[func Count\(s, substr string\) int](#)[func Cut\(s, sep string\) \(before, after string, found bool\)](#)[func CutPrefix\(s, prefix string\) \(after string, found bool\)](#)[func CutSuffix\(s, suffix string\) \(before string, found bool\)](#)[func EqualFold\(s, t string\) bool](#)[func Fields\(s string\) \[\]string](#)[func FieldsFunc\(s string, f func\(rune\) bool\) \[\]string](#)[func HasPrefix\(s, prefix string\) bool](#)[func HasSuffix\(s, suffix string\) bool](#)[func Index\(s, substr string\) int](#)[func IndexAny\(s, chars string\) int](#)[func IndexByte\(s string, c byte\) int](#)

```
func IndexFunc(s string, f func(rune) bool) int
func IndexRune(s string, r rune) int
func Join(elems []string, sep string) string
func LastIndex(s, substr string) int
func LastIndexAny(s, chars string) int
func LastIndexByte(s string, c byte) int
func LastIndexFunc(s string, f func(rune) bool) int
func Map(mapping func(rune) rune, s string) string
func Repeat(s string, count int) string
func Replace(s, old, new string, n int) string
func ReplaceAll(s, old, new string) string
func Split(s, sep string) []string
func SplitAfter(s, sep string) []string
func SplitAfterN(s, sep string, n int) []string
func SplitN(s, sep string, n int) []string
func Title(s string) string DEPRECATED
func ToLower(s string) string
func ToLowerSpecial(c unicode.SpecialCase, s string) string
func ToTitle(s string) string
func ToTitleSpecial(c unicode.SpecialCase, s string) string
func ToUpper(s string) string
func ToUpperSpecial(c unicode.SpecialCase, s string) string
func ToValidUTF8(s, replacement string) string
func Trim(s, cutset string) string
func TrimFunc(s string, f func(rune) bool) string
func TrimLeft(s, cutset string) string
func TrimLeftFunc(s string, f func(rune) bool) string
func TrimPrefix(s, prefix string) string
func TrimRight(s, cutset string) string
func TrimRightFunc(s string, f func(rune) bool) string
func TrimSpace(s string) string
func TrimSuffix(s, suffix string) string
type Builder
    func (b *Builder) Cap() int
    func (b *Builder) Grow(n int)
    func (b *Builder) Len() int
    func (b *Builder) Reset()
    func (b *Builder) String() string
    func (b *Builder) Write(p []byte) (int, error)
    func (b *Builder) WriteByte(c byte) error
    func (b *Builder) WriteRune(r rune) (int, error)
    func (b *Builder) WriteString(s string) (int, error)
type Reader
    func NewReader(s string) *Reader
    func (r *Reader) Len() int
    func (r *Reader) Read(b []byte) (n int, err error)
```

```
func (r *Reader) ReadAt(b []byte, off int64) (n int, err error)
func (r *Reader) ReadByte() (byte, error)
func (r *Reader) ReadRune() (ch rune, size int, err error)
func (r *Reader) Reset(s string)
func (r *Reader) Seek(offset int64, whence int) (int64, error)
func (r *Reader) Size() int64
func (r *Reader) UnreadByte() error
func (r *Reader) UnreadRune() error
func (r *Reader) WriteTo(w io.Writer) (n int64, err error)
```

type Replacer

```
func NewReplacer(oldnew ...string) *Replacer
func (r *Replacer) Replace(s string) string
func (r *Replacer) WriteString(w io.Writer, s string) (n int, err error)
```

Examples

[Builder](#)

[Compare](#)

[Contains](#)

[ContainsAny](#)

[ContainsRune](#)

[Count](#)

[Cut](#)

[EqualFold](#)

[Fields](#)

[FieldsFunc](#)

[HasPrefix](#)

[HasSuffix](#)

[Index](#)

[IndexAny](#)

[IndexByte](#)

[IndexFunc](#)

[IndexRune](#)

[Join](#)

[LastIndex](#)

[LastIndexAny](#)

[LastIndexByte](#)

[LastIndexFunc](#)

[Map](#)

[NewReplacer](#)

[Repeat](#)

[Replace](#)

[ReplaceAll](#)

[Split](#)

[SplitAfter](#)

[SplitAfterN](#)

[SplitN](#)

[Title](#)
[ToLower](#)
[ToLowerSpecial](#)
[ToTitle](#)
[ToTitleSpecial](#)
[ToUpper](#)
[ToUpperSpecial](#)
[Trim](#)
[TrimFunc](#)
[TrimLeft](#)
[TrimLeftFunc](#)
[TrimPrefix](#)
[TrimRight](#)
[TrimRightFunc](#)
[TrimSpace](#)
[TrimSuffix](#)

Constants

This section is empty.

Variables

This section is empty.

Functions

func [Clone](#)

added in go1.18

```
func Clone(s string) string
```

Clone returns a fresh copy of s. It guarantees to make a copy of s into a new allocation, which can be important when retaining only a small substring of a much larger string. Using Clone can help such programs use less memory. Of course, since using Clone makes a copy, overuse of Clone can make programs use more memory. Clone should typically be used only rarely, and only when profiling indicates that it is needed. For strings of length zero the string "" will be returned and no allocation is made.

func [Compare](#)

added in go1.5

```
func Compare(a, b string) int
```

Compare returns an integer comparing two strings lexicographically. The result will be 0 if a == b, -1 if a < b, and +1 if a > b.

Compare is included only for symmetry with package bytes. It is usually clearer and always faster to use the built-in string comparison operators ==, <, >, and so on.

► [Example](#)

func Contains

```
func Contains(s, substr string) bool
```

Contains reports whether substr is within s.

► [Example](#)

func ContainsAny

```
func ContainsAny(s, chars string) bool
```

ContainsAny reports whether any Unicode code points in chars are within s.

► [Example](#)

func ContainsRune

```
func ContainsRune(s string, r rune) bool
```

ContainsRune reports whether the Unicode code point r is within s.

► [Example](#)

func Count

```
func Count(s, substr string) int
```

Count counts the number of non-overlapping instances of substr in s. If substr is an empty string, Count returns 1 + the number of Unicode code points in s.

► [Example](#)

func Cut

added in go1.18

```
func Cut(s, sep string) (before, after string, found bool)
```

Cut slices s around the first instance of sep, returning the text before and after sep. The found result reports whether sep appears in s. If sep does not appear in s, cut returns s, "", false.

► [Example](#)

func CutPrefix

added in go1.20

```
func CutPrefix(s, prefix string) (after string, found bool)
```

CutPrefix returns s without the provided leading prefix string and reports whether it found the prefix. If s doesn't start with prefix, CutPrefix returns s, false. If prefix is the empty string, CutPrefix returns s, true.

func CutSuffix

added in go1.20

```
func CutSuffix(s, suffix string) (before string, found bool)
```

CutSuffix returns s without the provided ending suffix string and reports whether it found the suffix. If s doesn't end with suffix, CutSuffix returns s, false. If suffix is the empty string, CutSuffix returns s, true.

func EqualFold

```
func EqualFold(s, t string) bool
```

EqualFold reports whether s and t, interpreted as UTF-8 strings, are equal under simple Unicode case-folding, which is a more general form of case-insensitivity.

► [Example](#)

func Fields

```
func Fields(s string) []string
```

Fields splits the string s around each instance of one or more consecutive white space characters, as defined by unicode.IsSpace, returning a slice of substrings of s or an empty slice if s contains only white space.

► [Example](#)

func FieldsFunc

```
func FieldsFunc(s string, f func(rune) bool) []string
```

FieldsFunc splits the string s at each run of Unicode code points c satisfying f(c) and returns an array of slices of s. If all code points in s satisfy f(c) or the string is empty, an empty slice is returned.

FieldsFunc makes no guarantees about the order in which it calls f(c) and assumes that f always returns the same value for a given c.

► [Example](#)

func HasPrefix

```
func HasPrefix(s, prefix string) bool
```

HasPrefix tests whether the string `s` begins with `prefix`.

► [Example](#)

func HasSuffix

```
func HasSuffix(s, suffix string) bool
```

HasSuffix tests whether the string `s` ends with `suffix`.

► [Example](#)

func Index

```
func Index(s, substr string) int
```

Index returns the index of the first instance of `substr` in `s`, or -1 if `substr` is not present in `s`.

► [Example](#)

func IndexAny

```
func IndexAny(s, chars string) int
```

IndexAny returns the index of the first instance of any Unicode code point from `chars` in `s`, or -1 if no Unicode code point from `chars` is present in `s`.

► [Example](#)

func IndexByte

added in go1.2

```
func IndexByte(s string, c byte) int
```

IndexByte returns the index of the first instance of `c` in `s`, or -1 if `c` is not present in `s`.

► [Example](#)

func IndexFunc

```
func IndexFunc(s string, f func(rune) bool) int
```

IndexFunc returns the index into s of the first Unicode code point satisfying f(c), or -1 if none do.

► [Example](#)

func IndexRune

```
func IndexRune(s string, r rune) int
```

IndexRune returns the index of the first instance of the Unicode code point r, or -1 if rune is not present in s. If r is utf8.RuneError, it returns the first instance of any invalid UTF-8 byte sequence.

► [Example](#)

func Join

```
func Join(elems []string, sep string) string
```

Join concatenates the elements of its first argument to create a single string. The separator string sep is placed between elements in the resulting string.

► [Example](#)

func LastIndex

```
func LastIndex(s, substr string) int
```

LastIndex returns the index of the last instance of substr in s, or -1 if substr is not present in s.

► [Example](#)

func LastIndexAny

```
func LastIndexAny(s, chars string) int
```

LastIndexAny returns the index of the last instance of any Unicode code point from chars in s, or -1 if no Unicode code point from chars is present in s.

► [Example](#)

func LastIndexByte

added in go1.5

```
func LastIndexByte(s string, c byte) int
```

LastIndexByte returns the index of the last instance of c in s, or -1 if c is not present in s.

► [Example](#)

func **LastIndexFunc**

```
func LastIndexFunc(s string, f func(rune) bool) int
```

LastIndexFunc returns the index into s of the last Unicode code point satisfying f(c), or -1 if none do.

► [Example](#)

func **Map**

```
func Map(mapping func(rune) rune, s string) string
```

Map returns a copy of the string s with all its characters modified according to the mapping function. If mapping returns a negative value, the character is dropped from the string with no replacement.

► [Example](#)

func **Repeat**

```
func Repeat(s string, count int) string
```

Repeat returns a new string consisting of count copies of the string s.

It panics if count is negative or if the result of $(\text{len}(s) * \text{count})$ overflows.

► [Example](#)

func **Replace**

```
func Replace(s, old, new string, n int) string
```

Replace returns a copy of the string s with the first n non-overlapping instances of old replaced by new. If old is empty, it matches at the beginning of the string and after each UTF-8 sequence, yielding up to k+1 replacements for a k-rune string. If $n < 0$, there is no limit on the number of replacements.

► [Example](#)

func **ReplaceAll**

added in go1.12

```
func ReplaceAll(s, old, new string) string
```

ReplaceAll returns a copy of the string s with all non-overlapping instances of old replaced by new. If old is empty, it matches at the beginning of the string and after each UTF-8 sequence, yielding up to k+1 replacements for a k-rune string.

► [Example](#)

func Split

```
func Split(s, sep string) []string
```

Split slices s into all substrings separated by sep and returns a slice of the substrings between those separators.

If s does not contain sep and sep is not empty, Split returns a slice of length 1 whose only element is s.

If sep is empty, Split splits after each UTF-8 sequence. If both s and sep are empty, Split returns an empty slice.

It is equivalent to SplitN with a count of -1.

To split around the first instance of a separator, see Cut.

► [Example](#)

func SplitAfter

```
func SplitAfter(s, sep string) []string
```

SplitAfter slices s into all substrings after each instance of sep and returns a slice of those substrings.

If s does not contain sep and sep is not empty, SplitAfter returns a slice of length 1 whose only element is s.

If sep is empty, SplitAfter splits after each UTF-8 sequence. If both s and sep are empty, SplitAfter returns an empty slice.

It is equivalent to SplitAfterN with a count of -1.

► [Example](#)

func SplitAfterN

```
func SplitAfterN(s, sep string, n int) []string
```

SplitAfterN slices s into substrings after each instance of sep and returns a slice of those substrings.

The count determines the number of substrings to return:

```
n > 0: at most n substrings; the last substring will be the unsplit remainder.  
n == 0: the result is nil (zero substrings)  
n < 0: all substrings
```

Edge cases for `s` and `sep` (for example, empty strings) are handled as described in the documentation for `SplitAfter`.

► [Example](#)

func **SplitN**

```
func SplitN(s, sep string, n int) []string
```

`SplitN` slices `s` into substrings separated by `sep` and returns a slice of the substrings between those separators.

The count determines the number of substrings to return:

```
n > 0: at most n substrings; the last substring will be the unsplit remainder.  
n == 0: the result is nil (zero substrings)  
n < 0: all substrings
```

Edge cases for `s` and `sep` (for example, empty strings) are handled as described in the documentation for `Split`.

To split around the first instance of a separator, see `Cut`.

► [Example](#)

func **Title** DEPRECATED [Show](#)

func **ToLower**

```
func ToLower(s string) string
```

`ToLower` returns `s` with all Unicode letters mapped to their lower case.

► [Example](#)

func **ToLowerSpecial**

```
func ToLowerSpecial(c unicode.SpecialCase, s string) string
```

`ToLowerSpecial` returns a copy of the string `s` with all Unicode letters mapped to their lower case using the case mapping specified by `c`.

► [Example](#)

func [ToTitle](#)

```
func ToTitle(s string) string
```

ToTitle returns a copy of the string s with all Unicode letters mapped to their Unicode title case.

► [Example](#)

func [ToTitleSpecial](#)

```
func ToTitleSpecial(c unicode.SpecialCase, s string) string
```

ToTitleSpecial returns a copy of the string s with all Unicode letters mapped to their Unicode title case, giving priority to the special casing rules.

► [Example](#)

func [ToUpper](#)

```
func ToUpper(s string) string
```

ToUpper returns s with all Unicode letters mapped to their upper case.

► [Example](#)

func [ToUpperSpecial](#)

```
func ToUpperSpecial(c unicode.SpecialCase, s string) string
```

ToUpperSpecial returns a copy of the string s with all Unicode letters mapped to their upper case using the case mapping specified by c.

► [Example](#)

func [ToValidUTF8](#)

added in go1.13

```
func ToValidUTF8(s, replacement string) string
```

ToValidUTF8 returns a copy of the string s with each run of invalid UTF-8 byte sequences replaced by the replacement string, which may be empty.

func [Trim](#)

```
func Trim(s, cutset string) string
```

Trim returns a slice of the string `s` with all leading and trailing Unicode code points contained in `cutset` removed.

► [Example](#)

func TrimFunc

```
func TrimFunc(s string, f func(rune) bool) string
```

TrimFunc returns a slice of the string `s` with all leading and trailing Unicode code points `c` satisfying `f(c)` removed.

► [Example](#)

func TrimLeft

```
func TrimLeft(s, cutset string) string
```

TrimLeft returns a slice of the string `s` with all leading Unicode code points contained in `cutset` removed.

To remove a prefix, use `TrimPrefix` instead.

► [Example](#)

func TrimLeftFunc

```
func TrimLeftFunc(s string, f func(rune) bool) string
```

TrimLeftFunc returns a slice of the string `s` with all leading Unicode code points `c` satisfying `f(c)` removed.

► [Example](#)

func TrimPrefix

added in go1.1

```
func TrimPrefix(s, prefix string) string
```

TrimPrefix returns `s` without the provided leading prefix string. If `s` doesn't start with `prefix`, `s` is returned unchanged.

► [Example](#)

func TrimRight

```
func TrimRight(s, cutset string) string
```

TrimRight returns a slice of the string s, with all trailing Unicode code points contained in cutset removed.

To remove a suffix, use TrimSuffix instead.

► [Example](#)

func TrimRightFunc

```
func TrimRightFunc(s string, f func(rune) bool) string
```

TrimRightFunc returns a slice of the string s with all trailing Unicode code points c satisfying f(c) removed.

► [Example](#)

func TrimSpace

```
func TrimSpace(s string) string
```

TrimSpace returns a slice of the string s, with all leading and trailing white space removed, as defined by Unicode.

► [Example](#)

func TrimSuffix

added in go1.1

```
func TrimSuffix(s, suffix string) string
```

TrimSuffix returns s without the provided trailing suffix string. If s doesn't end with suffix, s is returned unchanged.

► [Example](#)

Types

type Builder

added in go1.10

```
type Builder struct {  
    // contains filtered or unexported fields  
}
```

A Builder is used to efficiently build a string using Write methods. It minimizes memory copying. The zero value is ready to use. Do not copy a non-zero Builder.

► Example

func (*Builder) Cap

added in go1.12

```
func (b *Builder) Cap() int
```

Cap returns the capacity of the builder's underlying byte slice. It is the total space allocated for the string being built and includes any bytes already written.

func (*Builder) Grow

added in go1.10

```
func (b *Builder) Grow(n int)
```

Grow grows b's capacity, if necessary, to guarantee space for another n bytes. After Grow(n), at least n bytes can be written to b without another allocation. If n is negative, Grow panics.

func (*Builder) Len

added in go1.10

```
func (b *Builder) Len() int
```

Len returns the number of accumulated bytes; b.Len() == len(b.String()).

func (*Builder) Reset

added in go1.10

```
func (b *Builder) Reset()
```

Reset resets the Builder to be empty.

func (*Builder) String

added in go1.10

```
func (b *Builder) String() string
```

String returns the accumulated string.

func (*Builder) Write

added in go1.10

```
func (b *Builder) Write(p []byte) (int, error)
```

Write appends the contents of p to b's buffer. Write always returns len(p), nil.

func (*Builder) WriteByte

added in go1.10

```
func (b *Builder) WriteByte(c byte) error
```

WriteByte appends the byte c to b's buffer. The returned error is always nil.

func (*Builder) WriteRune

added in go1.10

```
func (b *Builder) WriteRune(r rune) (int, error)
```

WriteRune appends the UTF-8 encoding of Unicode code point r to b's buffer. It returns the length of r and a nil error.

func (*Builder) WriteString

added in go1.10

```
func (b *Builder) WriteString(s string) (int, error)
```

WriteString appends the contents of s to b's buffer. It returns the length of s and a nil error.

type Reader

```
type Reader struct {  
    // contains filtered or unexported fields  
}
```

A Reader implements the io.Reader, io.ReaderAt, io.ByteReader, io.ByteScanner, io.RuneReader, io.RuneScanner, io.Seeker, and io.WriterTo interfaces by reading from a string. The zero value for Reader operates like a Reader of an empty string.

func NewReader

```
func NewReader(s string) *Reader
```

NewReader returns a new Reader reading from s. It is similar to bytes.NewBufferString but more efficient and read-only.

func (*Reader) Len

```
func (r *Reader) Len() int
```

Len returns the number of bytes of the unread portion of the string.

func (*Reader) Read

```
func (r *Reader) Read(b []byte) (n int, err error)
```

Read implements the io.Reader interface.

func (*Reader) ReadAt

```
func (r *Reader) ReadAt(b []byte, off int64) (n int, err error)
```


ReadAt implements the io.ReaderAt interface.

func (*Reader) ReadByte

```
func (r *Reader) ReadByte() (byte, error)
```

ReadByte implements the io.ByteReader interface.

func (*Reader) ReadRune

```
func (r *Reader) ReadRune() (ch rune, size int, err error)
```

ReadRune implements the io.RuneReader interface.

func (*Reader) Reset

added in go1.7

```
func (r *Reader) Reset(s string)
```

Reset resets the Reader to be reading from s.

func (*Reader) Seek

```
func (r *Reader) Seek(offset int64, whence int) (int64, error)
```

Seek implements the io.Seeker interface.

func (*Reader) Size

added in go1.5

```
func (r *Reader) Size() int64
```

Size returns the original length of the underlying string. Size is the number of bytes available for reading via ReadAt. The returned value is always the same and is not affected by calls to any other method.

func (*Reader) UnreadByte

```
func (r *Reader) UnreadByte() error
```

UnreadByte implements the io.ByteScanner interface.

func (*Reader) UnreadRune

```
func (r *Reader) UnreadRune() error
```

UnreadRune implements the io.RuneScanner interface.

func (*Reader) WriteTo

added in go1.1

```
func (r *Reader) WriteTo(w io.Writer) (n int64, err error)
```

WriteTo implements the io.WriterTo interface.

type Replacer

```
type Replacer struct {  
    // contains filtered or unexported fields  
}
```

Replacer replaces a list of strings with replacements. It is safe for concurrent use by multiple goroutines.

func NewReplacer

```
func NewReplacer(oldnew ...string) *Replacer
```

NewReplacer returns a new Replacer from a list of old, new string pairs. Replacements are performed in the order they appear in the target string, without overlapping matches. The old string comparisons are done in argument order.

NewReplacer panics if given an odd number of arguments.

► [Example](#)

func (*Replacer) Replace

```
func (r *Replacer) Replace(s string) string
```

Replace returns a copy of s with all replacements performed.

func (*Replacer) WriteString

```
func (r *Replacer) WriteString(w io.Writer, s string) (n int, err error)
```

WriteString writes s to w with all replacements performed.



Source Files

[View all](#)

[builder.go](#)
[clone.go](#)
[compare.go](#)

[reader.go](#)
[replace.go](#)
[search.go](#)

[strings.go](#)

Why Go

Use Cases

Case Studies

Get Started

Playground

Tour

Stack Overflow

Packages

Standard Library

About Go Packages

About

Download

Blog

Issue Tracker

[Help](#)

[Release Notes](#)

[Brand Guidelines](#)

[Code of Conduct](#)

Connect

[Twitter](#)

[GitHub](#)

[Slack](#)

[r/golang](#)

[Meetup](#)

[Golang Weekly](#)

[Copyright](#)

[Terms of Service](#)

[Privacy Policy](#)

[Report an Issue](#)



[Google](#)