utf8 package standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 0 | Imported by: 69,279

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 utf8 implements functions and constants to support text encoded in UTF-8. It includes functions to translate between runes and UTF-8 byte sequences. See https://en.wikipedia.org/wiki/UTF-8

Index

Constants
func AppendRune(p []byte, r rune) []byte
func DecodeLastRune(p []byte) (r rune, size int)
func DecodeLastRuneInString(s string) (r rune, size int)
func DecodeRune(p []byte) (r rune, size int)
func DecodeRuneInString(s string) (r rune, size int)
func EncodeRune(p []byte, r rune) int
func FullRune(p []byte) bool
func FullRuneInString(s string) bool
func RuneCount(p []byte) int
func RuneCountInString(s string) (n int)
func RuneLen(r rune) int
func RuneStart(b byte) bool
func Valid(p []byte) bool
func ValidRune(r rune) bool
func ValidString(s string) bool

Examples

AppendRune
DecodeLastRune
DecodeLastRuneInString
DecodeRune
DecodeRuneInString
EncodeRune
EncodeRune (OutOfRange)
FullRune
FullRuneInString
RuneCount
RuneCountInString
RuneLen
RuneStart
Valid
ValidRune
ValidString

Constants

View Source

```
const (
    RuneError = '\uFFFD'     // the "error" Rune or "Unicode replacement character"
    RuneSelf  = 0x80         // characters below RuneSelf are represented as themselves
    MaxRune   = '\U0010FFFF' // Maximum valid Unicode code point.
    UTFMax    = 4            // maximum number of bytes of a UTF-8 encoded Unicode chara
)
```

Numbers fundamental to the encoding.

Variables

This section is empty.

Functions

func AppendRune

added in go1.18

```
func AppendRune(p []byte, r rune) []byte
```

AppendRune appends the UTF-8 encoding of r to the end of p and returns the extended buffer. If the rune is out of range, it appends the encoding of RuneError.

▶ Example

func DecodeLastRune

```
func DecodeLastRune(p []byte) (r rune, size int)
```

DecodeLastRune unpacks the last UTF-8 encoding in p and returns the rune and its width in bytes. If p is empty it returns (RuneError, 0). Otherwise, if the encoding is invalid, it returns (RuneError, 1). Both are impossible results for correct, non-empty UTF-8.

An encoding is invalid if it is incorrect UTF-8, encodes a rune that is out of range, or is not the shortest possible UTF-8 encoding for the value. No other validation is performed.

▶ Example

func DecodeLastRuneInString

```
func DecodeLastRuneInString(s string) (r rune, size int)
```

DecodeLastRuneInString is like DecodeLastRune but its input is a string. If s is empty it returns (RuneError, 0). Otherwise, if the encoding is invalid, it returns (RuneError, 1). Both are impossible results for correct, non-empty UTF-8.

An encoding is invalid if it is incorrect UTF-8, encodes a rune that is out of range, or is not the shortest possible UTF-8 encoding for the value. No other validation is performed.

▶ Example

func DecodeRune

```
func DecodeRune(p []byte) (r rune, size int)
```

DecodeRune unpacks the first UTF-8 encoding in p and returns the rune and its width in bytes. If p is empty it returns (RuneError, 0). Otherwise, if the encoding is invalid, it returns (RuneError, 1). Both are impossible results for correct, non-empty UTF-8.

An encoding is invalid if it is incorrect UTF-8, encodes a rune that is out of range, or is not the shortest possible UTF-8 encoding for the value. No other validation is performed.

▶ Example

func DecodeRuneInString

```
func DecodeRuneInString(s string) (r rune, size int)
```

DecodeRuneInString is like DecodeRune but its input is a string. If s is empty it returns (RuneError, 0). Otherwise, if the encoding is invalid, it returns (RuneError, 1). Both are impossible results for correct, non-empty UTF-8.

An encoding is invalid if it is incorrect UTF-8, encodes a rune that is out of range, or is not the shortest possible UTF-8 encoding for the value. No other validation is performed.

▶ Example

func EncodeRune

```
func EncodeRune(p []byte, r rune) int
```

EncodeRune writes into p (which must be large enough) the UTF-8 encoding of the rune. If the rune is out of range, it writes the encoding of RuneError. It returns the number of bytes written.

▶ Example

▶ Example (OutOfRange)

func FullRune

```
func FullRune(p []byte) bool
```

FullRune reports whether the bytes in p begin with a full UTF-8 encoding of a rune. An invalid encoding is considered a full Rune since it will convert as a width-1 error rune.

▶ Example

func FullRuneInString

```
func FullRuneInString(s string) bool
```

FullRuneInString is like FullRune but its input is a string.

▶ Example

func RuneCount

```
func RuneCount(p []byte) int
```

RuneCount returns the number of runes in p. Erroneous and short encodings are treated as single runes of width 1 byte.

▶ Example

func RuneCountInString

```
func RuneCountInString(s string) (n int)
```

RuneCountInString is like RuneCount but its input is a string.

▶ Example

func RuneLen

```
func RuneLen(r rune) int
```

RuneLen returns the number of bytes required to encode the rune. It returns -1 if the rune is not a valid value to encode in UTF-8.

▶ Example

func RuneStart

```
func RuneStart(b byte) bool
```

RuneStart reports whether the byte could be the first byte of an encoded, possibly invalid rune. Second and subsequent bytes always have the top two bits set to 10.

▶ Example

func Valid

```
func Valid(p []byte) bool
```

Valid reports whether p consists entirely of valid UTF-8-encoded runes.

▶ Example

func ValidRune

added in go1.1

```
func ValidRune(r rune) bool
```

ValidRune reports whether r can be legally encoded as UTF-8. Code points that are out of range or a surrogate half are illegal.

▶ Example

func ValidString

```
func ValidString(s string) bool
```

ValidString reports whether s consists entirely of valid UTF-8-encoded runes.

▶ Example

Types

This section is empty.

Source Files

View all

utf8.go