unicode

package standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 0 |
Imported by: 65,848

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 unicode provides data and functions to test some properties of Unicode code points.

▶ Example (Is)

Index

Examples

Constants

View Source

```
const (
    MaxRune         = '\U0010FFFF' // Maximum valid Unicode code point.
    ReplacementChar = '\uFFFD'     // Represents invalid code points.
    MaxASCII        = '\u007F'     // maximum ASCII value.
    MaxLatin1       = '\u00FF'     // maximum Latin-1 value.
)
```

View Source

```
const (
    UpperCase = iota
    LowerCase
    TitleCase
    MaxCase
)
```

Indices into the Delta arrays inside CaseRanges for case mapping.

View Source

```
const (
    UpperLower = MaxRune + 1 // (Cannot be a valid delta.)
)
```

If the Delta field of a CaseRange is UpperLower, it means this CaseRange represents a sequence of the form (say) Upper Lower Upper Lower.

View Source

```
const Version = "13.0.0"
```

Version is the Unicode edition from which the tables are derived.

Variables

View Source

```
var (
    Cc     = _Cc // Cc is the set of Unicode characters in category Cc (Other, control)
    Cf     = _Cf // Cf is the set of Unicode characters in category Cf (Other, format).
    Co     = _Co // Co is the set of Unicode characters in category Co (Other, private 
    Cs     = _Cs // Cs is the set of Unicode characters in category Cs (Other, surrogat
    Digit  = _Nd // Digit is the set of Unicode characters with the "decimal digit" pro
    Nd     = _Nd // Nd is the set of Unicode characters in category Nd (Number, decimal
    Letter = _L  // Letter/L is the set of Unicode letters, category L.
    L      = _L
    Lm     = _Lm // Lm is the set of Unicode characters in category Lm (Letter, modifie
    Lo     = _Lo // Lo is the set of Unicode characters in category Lo (Letter, other).
    Lower  = _Ll // Lower is the set of Unicode lower case letters.
    Ll     = _Ll // Ll is the set of Unicode characters in category Ll (Letter, lowerca
    Mark   = _M  // Mark/M is the set of Unicode mark characters, category M.
    M      = _M
    Mc     = _Mc // Mc is the set of Unicode characters in category Mc (Mark, spacing c
    Me     = _Me // Me is the set of Unicode characters in category Me (Mark, enclosing
    Mn     = _Mn // Mn is the set of Unicode characters in category Mn (Mark, nonspacin
    Nl     = _Nl // Nl is the set of Unicode characters in category Nl (Number, letter)
    No     = _No // No is the set of Unicode characters in category No (Number, other).
    Number = _N  // Number/N is the set of Unicode number characters, category N.
    N      = _N
    Other  = _C // Other/C is the set of Unicode control and special characters, catego
    C      = _C
    Pc     = _Pc // Pc is the set of Unicode characters in category Pc (Punctuation, co
    Pd     = _Pd // Pd is the set of Unicode characters in category Pd (Punctuation, da
    Pe     = _Pe // Pe is the set of Unicode characters in category Pe (Punctuation, cl
    Pf     = _Pf // Pf is the set of Unicode characters in category Pf (Punctuation, fi
    Pi     = _Pi // Pi is the set of Unicode characters in category Pi (Punctuation, in
    Po     = _Po // Po is the set of Unicode characters in category Po (Punctuation, oth
    Ps     = _Ps // Ps is the set of Unicode characters in category Ps (Punctuation, ope
    Punct  = _P  // Punct/P is the set of Unicode punctuation characters, category P.
    P      = _P
    Sc     = _Sc // Sc is the set of Unicode characters in category Sc (Symbol, currency
    Sk     = _Sk // Sk is the set of Unicode characters in category Sk (Symbol, modifie
    Sm     = _Sm // Sm is the set of Unicode characters in category Sm (Symbol, math).
    So     = _So // So is the set of Unicode characters in category So (Symbol, other).
    Space  = _Z  // Space/Z is the set of Unicode space characters, category Z.
    Z      = _Z
    Symbol = _S // Symbol/S is the set of Unicode symbol characters, category S.
    S      = _S
    Title  = _Lt // Title is the set of Unicode title case letters.
    Lt     = _Lt // Lt is the set of Unicode characters in category Lt (Letter, titlecas
    Upper  = _Lu // Upper is the set of Unicode upper case letters.
    Lu     = _Lu // Lu is the set of Unicode characters in category Lu (Letter, uppercas
    Zl     = _Zl // Zl is the set of Unicode characters in category Zl (Separator, line
    Zp     = _Zp // Zp is the set of Unicode characters in category Zp (Separator, parag
    Zs     = _Zs // Zs is the set of Unicode characters in category Zs (Separator, space
)
```

These variables have type *RangeTable.

View Source

```
var (
    Adlam                  = _Adlam                  // Adlam is the set of Unicode cha
    Ahom                   = _Ahom                   // Ahom is the set of Unicode chara
    Anatolian_Hieroglyphs  = _Anatolian_Hieroglyphs  // Anatolian_Hieroglyphs is the se
    Arabic                 = _Arabic                 // Arabic is the set of Unicode cha
    Armenian               = _Armenian               // Armenian is the set of Unicode c
    Avestan                = _Avestan                // Avestan is the set of Unicode ch
    Balinese               = _Balinese               // Balinese is the set of Unicode c
    Bamum                  = _Bamum                  // Bamum is the set of Unicode char
    Bassa_Vah              = _Bassa_Vah              // Bassa_Vah is the set of Unicode
    Batak                  = _Batak                  // Batak is the set of Unicode char
    Bengali                = _Bengali                // Bengali is the set of Unicode ch
    Bhaiksuki              = _Bhaiksuki              // Bhaiksuki is the set of Unicode
    Bopomofo               = _Bopomofo               // Bopomofo is the set of Unicode c
    Brahmi                 = _Brahmi                 // Brahmi is the set of Unicode cha
    Braille                = _Braille                // Braille is the set of Unicode ch
    Buginese               = _Buginese               // Buginese is the set of Unicode c
    Buhid                  = _Buhid                  // Buhid is the set of Unicode char
    Canadian_Aboriginal    = _Canadian_Aboriginal    // Canadian_Aboriginal is the set c
    Carian                 = _Carian                 // Carian is the set of Unicode cha
    Caucasian_Albanian     = _Caucasian_Albanian     // Caucasian_Albanian is the set of
    Chakma                 = _Chakma                 // Chakma is the set of Unicode cha
    Cham                   = _Cham                   // Cham is the set of Unicode chara
    Cherokee               = _Cherokee               // Cherokee is the set of Unicode c
    Chorasmian             = _Chorasmian             // Chorasmian is the set of Unicode
    Common                 = _Common                 // Common is the set of Unicode cha
    Coptic                 = _Coptic                 // Coptic is the set of Unicode cha
    Cuneiform              = _Cuneiform              // Cuneiform is the set of Unicode
    Cypriot                = _Cypriot                // Cypriot is the set of Unicode cl
    Cyrillic               = _Cyrillic               // Cyrillic is the set of Unicode c
    Deseret                = _Deseret                // Deseret is the set of Unicode cl
    Devanagari             = _Devanagari             // Devanagari is the set of Unicode
    Dives_Akuru            = _Dives_Akuru            // Dives_Akuru is the set of Unicod
    Dogra                  = _Dogra                  // Dogra is the set of Unicode char
    Duployan               = _Duployan               // Duployan is the set of Unicode c
    Egyptian_Hieroglyphs   = _Egyptian_Hieroglyphs   // Egyptian_Hieroglyphs is the set
    Elbasan                = _Elbasan                // Elbasan is the set of Unicode cl
    Elymaic                = _Elymaic                // Elymaic is the set of Unicode cl
    Ethiopic               = _Ethiopic               // Ethiopic is the set of Unicode c
    Georgian               = _Georgian               // Georgian is the set of Unicode c
    Glagolitic             = _Glagolitic             // Glagolitic is the set of Unicode
    Gothic                 = _Gothic                 // Gothic is the set of Unicode cha
    Grantha                = _Grantha                // Grantha is the set of Unicode cl
    Greek                  = _Greek                  // Greek is the set of Unicode char
    Gujarati               = _Gujarati               // Gujarati is the set of Unicode c
    Gunjala_Gondi          = _Gunjala_Gondi          // Gunjala_Gondi is the set of Unic
    Gurmukhi               = _Gurmukhi               // Gurmukhi is the set of Unicode c
    Han                    = _Han                    // Han is the set of Unicode charac
    Hangul                 = _Hangul                 // Hangul is the set of Unicode cha
    Hanifi_Rohingya        = _Hanifi_Rohingya        // Hanifi_Rohingya is the set of Un
    Hanunoo                = _Hanunoo                // Hanunoo is the set of Unicode cl
    Hatran                 = _Hatran                 // Hatran is the set of Unicode cha
    Hebrew                 = _Hebrew                 // Hebrew is the set of Unicode cha
    Hiragana               = _Hiragana               // Hiragana is the set of Unicode c
    Imperial_Aramaic       = _Imperial_Aramaic       // Imperial_Aramaic is the set of U
    Inherited              = _Inherited              // Inherited is the set of Unicode
    Inscriptional_Pahlavi  = _Inscriptional_Pahlavi  // Inscriptional_Pahlavi is the set
    Inscriptional_Parthian = _Inscriptional_Parthian // Inscriptional_Parthian is the se
    Javanese               = _Javanese               // Javanese is the set of Unicode c
    Kaithi                 = _Kaithi                 // Kaithi is the set of Unicode cha
    Kannada                = _Kannada                // Kannada is the set of Unicode cl
    Katakana               = _Katakana               // Katakana is the set of Unicode c
    Kayah_Li               = _Kayah_Li               // Kayah_Li is the set of Unicode c
    Kharoshthi             = _Kharoshthi             // Kharoshthi is the set of Unicode
    Khitan_Small_Script    = _Khitan_Small_Script    // Khitan_Small_Script is the set o
    Khmer                  = _Khmer                  // Khmer is the set of Unicode char
    Khojki                 = _Khojki                 // Khojki is the set of Unicode cha
    Khudawadi              = _Khudawadi              // Khudawadi is the set of Unicode
    Lao                    = _Lao                    // Lao is the set of Unicode charac
    Latin                  = _Latin                  // Latin is the set of Unicode char
    Lepcha                 = _Lepcha                 // Lepcha is the set of Unicode cha
    Limbu                  = _Limbu                  // Limbu is the set of Unicode char
    Linear_A               = _Linear_A               // Linear_A is the set of Unicode c
    Linear_B               = _Linear_B               // Linear_B is the set of Unicode c
    Lisu                   = _Lisu                   // Lisu is the set of Unicode chara
    Lycian                 = _Lycian                 // Lycian is the set of Unicode cha
    Lydian                 = _Lydian                 // Lydian is the set of Unicode cha
    Mahajani               = _Mahajani               // Mahajani is the set of Unicode c
    Makasar                = _Makasar                // Makasar is the set of Unicode cl
    Malayalam              = _Malayalam              // Malayalam is the set of Unicode
    Mandaic                = _Mandaic                // Mandaic is the set of Unicode cl
    Manichaean             = _Manichaean             // Manichaean is the set of Unicode
    Marchen                = _Marchen                // Marchen is the set of Unicode cl
    Masaram_Gondi          = _Masaram_Gondi          // Masaram_Gondi is the set of Unic
    Medefaidrin            = _Medefaidrin            // Medefaidrin is the set of Unicod
    Meetei_Mayek           = _Meetei_Mayek           // Meetei_Mayek is the set of Unic
    Mende_Kikakui          = _Mende_Kikakui          // Mende_Kikakui is the set of Unic
    Meroitic_Cursive       = _Meroitic_Cursive       // Meroitic_Cursive is the set of U
    Meroitic_Hieroglyphs   = _Meroitic_Hieroglyphs   // Meroitic_Hieroglyphs is the set
    Miao                   = _Miao                   // Miao is the set of Unicode chara
    Modi                   = _Modi                   // Modi is the set of Unicode chara
    Mongolian              = _Mongolian              // Mongolian is the set of Unicode
    Mro                    = _Mro                    // Mro is the set of Unicode charac
    Multani                = _Multani                // Multani is the set of Unicode cl
    Myanmar                = _Myanmar                // Myanmar is the set of Unicode cl
    Nabataean              = _Nabataean              // Nabataean is the set of Unicode
    Nandinagari            = _Nandinagari            // Nandinagari is the set of Unicod
    New_Tai_Lue            = _New_Tai_Lue            // New_Tai_Lue is the set of Unicod
    Newa                   = _Newa                   // Newa is the set of Unicode chara
    Nko                    = _Nko                    // Nko is the set of Unicode charac
    Nushu                  = _Nushu                  // Nushu is the set of Unicode cha
    Nyiakeng_Puachue_Hmong = _Nyiakeng_Puachue_Hmong // Nyiakeng_Puachue_Hmong is the se
    Ogham                  = _Ogham                  // Ogham is the set of Unicode cha
    Ol_Chiki               = _Ol_Chiki               // Ol_Chiki is the set of Unicode c
    Old_Hungarian          = _Old_Hungarian          // Old_Hungarian is the set of Unic
    Old_Italic             = _Old_Italic             // Old_Italic is the set of Unicode
    Old_North_Arabian      = _Old_North_Arabian      // Old_North_Arabian is the set of
    Old_Permic             = _Old_Permic             // Old_Permic is the set of Unicode
    Old_Persian            = _Old_Persian            // Old_Persian is the set of Unicod
    Old_Sogdian            = _Old_Sogdian            // Old_Sogdian is the set of Unicod
    Old_South_Arabian      = _Old_South_Arabian      // Old_South_Arabian is the set of
    Old_Turkic             = _Old_Turkic             // Old_Turkic is the set of Unicode
    Oriya                  = _Oriya                  // Oriya is the set of Unicode cha
    Osage                  = _Osage                  // Osage is the set of Unicode cha
    Osmanya                = _Osmanya                // Osmanya is the set of Unicode cl
    Pahawh_Hmong           = _Pahawh_Hmong           // Pahawh_Hmong is the set of Unic
    Palmyrene              = _Palmyrene              // Palmyrene is the set of Unicode
    Pau_Cin_Hau            = _Pau_Cin_Hau            // Pau_Cin_Hau is the set of Unicod
    Phags_Pa               = _Phags_Pa               // Phags_Pa is the set of Unicode c
    Phoenician             = _Phoenician             // Phoenician is the set of Unicode
    Psalter_Pahlavi        = _Psalter_Pahlavi        // Psalter_Pahlavi is the set of Un
    Rejang                 = _Rejang                 // Rejang is the set of Unicode cha
    Runic                  = _Runic                  // Runic is the set of Unicode cha
    Samaritan              = _Samaritan              // Samaritan is the set of Unicode
    Saurashtra             = _Saurashtra             // Saurashtra is the set of Unicode
    Sharada                = _Sharada                // Sharada is the set of Unicode cl
    Shavian                = _Shavian                // Shavian is the set of Unicode cl
    Siddham                = _Siddham                // Siddham is the set of Unicode cl
    SignWriting            = _SignWriting            // SignWriting is the set of Unicod
    Sinhala                = _Sinhala                // Sinhala is the set of Unicode cl
    Sogdian                = _Sogdian                // Sogdian is the set of Unicode cl
    Sora_Sompeng           = _Sora_Sompeng           // Sora_Sompeng is the set of Unic
    Soyombo                = _Soyombo                // Soyombo is the set of Unicode cl
    Sundanese              = _Sundanese              // Sundanese is the set of Unicode
    Syloti_Nagri           = _Syloti_Nagri           // Syloti_Nagri is the set of Unic
    Syriac                 = _Syriac                 // Syriac is the set of Unicode ch
    Tagalog                = _Tagalog                // Tagalog is the set of Unicode c
    Tagbanwa               = _Tagbanwa               // Tagbanwa is the set of Unicode 
    Tai_Le                 = _Tai_Le                 // Tai_Le is the set of Unicode ch
    Tai_Tham               = _Tai_Tham               // Tai_Tham is the set of Unicode 
    Tai_Viet               = _Tai_Viet               // Tai_Viet is the set of Unicode 
    Takri                  = _Takri                  // Takri is the set of Unicode cha
    Tamil                  = _Tamil                  // Tamil is the set of Unicode cha
    Tangut                 = _Tangut                 // Tangut is the set of Unicode ch
    Telugu                 = _Telugu                 // Telugu is the set of Unicode ch
    Thaana                 = _Thaana                 // Thaana is the set of Unicode ch
    Thai                   = _Thai                   // Thai is the set of Unicode char
    Tibetan                = _Tibetan                // Tibetan is the set of Unicode c
    Tifinagh               = _Tifinagh               // Tifinagh is the set of Unicode 
    Tirhuta                = _Tirhuta                // Tirhuta is the set of Unicode c
    Ugaritic               = _Ugaritic               // Ugaritic is the set of Unicode 
    Vai                    = _Vai                    // Vai is the set of Unicode chara
    Wancho                 = _Wancho                 // Wancho is the set of Unicode ch
    Warang_Citi            = _Warang_Citi            // Warang_Citi is the set of Unico
    Yezidi                 = _Yezidi                 // Yezidi is the set of Unicode ch
    Yi                     = _Yi                     // Yi is the set of Unicode charac
    Zanabazar_Square       = _Zanabazar_Square       // Zanabazar_Square is the set of 
)
```

These variables have type *RangeTable.

View Source

```
var (
    ASCII_Hex_Digit                    = _ASCII_Hex_Digit                    // ASCII_He
    Bidi_Control                       = _Bidi_Control                       // Bidi_Co
    Dash                               = _Dash                               // Dash is
    Deprecated                         = _Deprecated                         // Depreca
    Diacritic                          = _Diacritic                          // Diacrit
    Extender                           = _Extender                           // Extende
    Hex_Digit                          = _Hex_Digit                          // Hex_Dig
    Hyphen                             = _Hyphen                             // Hyphen 
    IDS_Binary_Operator                = _IDS_Binary_Operator                // IDS_Bin
    IDS_Trinary_Operator               = _IDS_Trinary_Operator               // IDS_Tri
    Ideographic                        = _Ideographic                        // Ideogra
    Join_Control                       = _Join_Control                       // Join_Co
    Logical_Order_Exception            = _Logical_Order_Exception            // Logical
    Noncharacter_Code_Point            = _Noncharacter_Code_Point            // Nonchar
    Other_Alphabetic                   = _Other_Alphabetic                   // Other_A
    Other_Default_Ignorable_Code_Point = _Other_Default_Ignorable_Code_Point // Other_D
    Other_Grapheme_Extend              = _Other_Grapheme_Extend              // Other_G
    Other_ID_Continue                  = _Other_ID_Continue                  // Other_I
    Other_ID_Start                     = _Other_ID_Start                     // Other_I
    Other_Lowercase                    = _Other_Lowercase                    // Other_L
    Other_Math                         = _Other_Math                         // Other_M
    Other_Uppercase                    = _Other_Uppercase                    // Other_U
    Pattern_Syntax                     = _Pattern_Syntax                     // Pattern
    Pattern_White_Space                = _Pattern_White_Space               // Pattern_
    Prepended_Concatenation_Mark       = _Prepended_Concatenation_Mark      // Prepend
    Quotation_Mark                     = _Quotation_Mark                    // Quotati
    Radical                            = _Radical                           // Radical
    Regional_Indicator                 = _Regional_Indicator                // Regiona
    STerm                              = _Sentence_Terminal                 // STerm i
    Sentence_Terminal                  = _Sentence_Terminal                 // Sentenc
    Soft_Dotted                        = _Soft_Dotted                       // Soft_Do
    Terminal_Punctuation               = _Terminal_Punctuation              // Termina
    Unified_Ideograph                  = _Unified_Ideograph                 // Unified
    Variation_Selector                 = _Variation_Selector                // Variati
    White_Space                        = _White_Space                       // White_S
)
```

These variables have type *RangeTable.

View Source

```
var CaseRanges = _CaseRanges
```

CaseRanges is the table describing case mappings for all letters with non-self mappings.

View Source

```
var Categories = map[string]*RangeTable{
    "C":  C,
    "Cc": Cc,
    "Cf": Cf,
    "Co": Co,
    "Cs": Cs,
    "L":  L,
    "Ll": Ll,
    "Lm": Lm,
    "Lo": Lo,
    "Lt": Lt,
    "Lu": Lu,
    "M":  M,
    "Mc": Mc,
    "Me": Me,
    "Mn": Mn,
    "N":  N,
    "Nd": Nd,
    "Nl": Nl,
    "No": No,
    "P":  P,
    "Pc": Pc,
    "Pd": Pd,
    "Pe": Pe,
    "Pf": Pf,
    "Pi": Pi,
    "Po": Po,
    "Ps": Ps,
    "S":  S,
    "Sc": Sc,
    "Sk": Sk,
    "Sm": Sm,
    "So": So,
    "Z":  Z,
    "Zl": Zl,
    "Zp": Zp,
    "Zs": Zs,
}
```

Categories is the set of Unicode category tables.

View Source

```
var FoldCategory = map[string]*RangeTable{
    "L":  foldL,
    "Ll": foldLl,
    "Lt": foldLt,
    "Lu": foldLu,
    "M":  foldM,
    "Mn": foldMn,
}
```

FoldCategory maps a category name to a table of code points outside the category that are equivalent under simple case folding to code points inside the category. If there is no entry for a category name, there are no such points.

View Source

```
var FoldScript = map[string]*RangeTable{
    "Common":    foldCommon,
    "Greek":     foldGreek,
    "Inherited": foldInherited,
}
```

FoldScript maps a script name to a table of code points outside the script that are equivalent under simple case folding to code points inside the script. If there is no entry for a script name, there are no such points.

View Source

```
var GraphicRanges = []*RangeTable{
    L, M, N, P, S, Zs,
}
```

GraphicRanges defines the set of graphic characters according to Unicode.

View Source

```
var PrintRanges = []*RangeTable{
    L, M, N, P, S,
}
```

PrintRanges defines the set of printable characters according to Go. ASCII space, U+0020, is handled separately.

View Source

```
var Properties = map[string]*RangeTable{
	"ASCII_Hex_Digit":                    ASCII_Hex_Digit,
	"Bidi_Control":                       Bidi_Control,
	"Dash":                               Dash,
	"Deprecated":                         Deprecated,
	"Diacritic":                          Diacritic,
	"Extender":                           Extender,
	"Hex_Digit":                          Hex_Digit,
	"Hyphen":                             Hyphen,
	"IDS_Binary_Operator":                IDS_Binary_Operator,
	"IDS_Trinary_Operator":               IDS_Trinary_Operator,
	"Ideographic":                        Ideographic,
	"Join_Control":                       Join_Control,
	"Logical_Order_Exception":            Logical_Order_Exception,
	"Noncharacter_Code_Point":            Noncharacter_Code_Point,
	"Other_Alphabetic":                   Other_Alphabetic,
	"Other_Default_Ignorable_Code_Point": Other_Default_Ignorable_Code_Point,
	"Other_Grapheme_Extend":              Other_Grapheme_Extend,
	"Other_ID_Continue":                  Other_ID_Continue,
	"Other_ID_Start":                     Other_ID_Start,
	"Other_Lowercase":                    Other_Lowercase,
	"Other_Math":                         Other_Math,
	"Other_Uppercase":                    Other_Uppercase,
	"Pattern_Syntax":                     Pattern_Syntax,
	"Pattern_White_Space":                Pattern_White_Space,
	"Prepended_Concatenation_Mark":       Prepended_Concatenation_Mark,
	"Quotation_Mark":                     Quotation_Mark,
	"Radical":                            Radical,
	"Regional_Indicator":                 Regional_Indicator,
	"Sentence_Terminal":                  Sentence_Terminal,
	"STerm":                              Sentence_Terminal,
	"Soft_Dotted":                        Soft_Dotted,
	"Terminal_Punctuation":               Terminal_Punctuation,
	"Unified_Ideograph":                  Unified_Ideograph,
	"Variation_Selector":                 Variation_Selector,
	"White_Space":                        White_Space,
}
```

Properties is the set of Unicode property tables.

View Source

```
var Scripts = map[string]*RangeTable{}/* 156 elements not displayed */
```

Scripts is the set of Unicode script tables.

Functions

func In

added in go1.2

```
func In(r rune, ranges ...*RangeTable) bool
```

In reports whether the rune is a member of one of the ranges.

func Is

```
func Is(rangeTab *RangeTable, r rune) bool
```

Is reports whether the rune is in the specified table of ranges.

func IsControl

```
func IsControl(r rune) bool
```

IsControl reports whether the rune is a control character. The C (Other) Unicode category includes more code points such as surrogates; use Is(C, r) to test for them.

func IsDigit

```
func IsDigit(r rune) bool
```

IsDigit reports whether the rune is a decimal digit.

▶ Example

func IsGraphic

```
func IsGraphic(r rune) bool
```

IsGraphic reports whether the rune is defined as a Graphic by Unicode. Such characters include letters, marks, numbers, punctuation, symbols, and spaces, from categories L, M, N, P, S, Zs.

func IsLetter

```
func IsLetter(r rune) bool
```

IsLetter reports whether the rune is a letter (category L).

▶ Example

func IsLower

```
func IsLower(r rune) bool
```

IsLower reports whether the rune is a lower case letter.

▶ Example

func IsMark

```
func IsMark(r rune) bool
```

IsMark reports whether the rune is a mark character (category M).

func IsNumber

```
func IsNumber(r rune) bool
```

IsNumber reports whether the rune is a number (category N).

▶ Example

func IsOneOf

```
func IsOneOf(ranges []*RangeTable, r rune) bool
```

IsOneOf reports whether the rune is a member of one of the ranges. The function "In" provides a nicer signature and should be used in preference to IsOneOf.

func IsPrint

```
func IsPrint(r rune) bool
```

IsPrint reports whether the rune is defined as printable by Go. Such characters include letters, marks, numbers, punctuation, symbols, and the ASCII space character, from categories L, M, N, P, S and the ASCII space character. This categorization is the same as IsGraphic except that the only spacing character is ASCII space, U+0020.

func IsPunct

```
func IsPunct(r rune) bool
```

IsPunct reports whether the rune is a Unicode punctuation character (category P).

func IsSpace

```
func IsSpace(r rune) bool
```

IsSpace reports whether the rune is a space character as defined by Unicode's White Space property; in the Latin-1 space this is

```
'\t', '\n', '\v', '\f', '\r', ' ', U+0085 (NEL), U+00A0 (NBSP).
```

Other definitions of spacing characters are set by category Z and property Pattern_White_Space.

▶ Example

func IsSymbol

```
func IsSymbol(r rune) bool
```

IsSymbol reports whether the rune is a symbolic character.

func IsTitle

```
func IsTitle(r rune) bool
```

IsTitle reports whether the rune is a title case letter.

▶ Example

func IsUpper

```
func IsUpper(r rune) bool
```

IsUpper reports whether the rune is an upper case letter.

▶ Example

func SimpleFold

```
func SimpleFold(r rune) rune
```

SimpleFold iterates over Unicode code points equivalent under the Unicode-defined simple case folding. Among the code points equivalent to rune (including rune itself), SimpleFold returns the smallest rune > r if one exists, or else the smallest rune >= 0. If r is not a valid Unicode code point, SimpleFold(r) returns r.

For example:

```
SimpleFold('A') = 'a'
SimpleFold('a') = 'A'

SimpleFold('K') = 'k'
SimpleFold('k') = '\u212A' (Kelvin symbol, K)
SimpleFold('\u212A') = 'K'

SimpleFold('1') = '1'
SimpleFold(-2) = -2
```

▶ Example

func To

```
func To(_case int, r rune) rune
```

To maps the rune to the specified case: UpperCase, LowerCase, or TitleCase.

▶ Example

func ToLower

```
func ToLower(r rune) rune
```

ToLower maps the rune to lower case.

▶ Example

func ToTitle

```
func ToTitle(r rune) rune
```

ToTitle maps the rune to title case.

▶ Example

func ToUpper

```
func ToUpper(r rune) rune
```

ToUpper maps the rune to upper case.

▶ Example

Types

type CaseRange

```
type CaseRange struct {
    Lo    uint32
    Hi    uint32
    Delta d
}
```

CaseRange represents a range of Unicode code points for simple (one code point to one code point) case conversion. The range runs from Lo to Hi inclusive, with a fixed stride of 1. Deltas are the number to add to the code point to reach the code point for a different case for that character. They may be negative. If zero, it means the character is in the corresponding case. There is a special case representing sequences of alternating corresponding Upper and Lower pairs. It appears with a fixed Delta of

```
{UpperLower, UpperLower, UpperLower}
```

The constant UpperLower has an otherwise impossible delta value.

type Range16

```
type Range16 struct {
    Lo     uint16
    Hi     uint16
    Stride uint16
}
```

Range16 represents of a range of 16-bit Unicode code points. The range runs from Lo to Hi inclusive and has the specified stride.

type Range32

```
type Range32 struct {
    Lo     uint32
    Hi     uint32
    Stride uint32
}
```

Range32 represents of a range of Unicode code points and is used when one or more of the values will not fit in 16 bits. The range runs from Lo to Hi inclusive and has the specified stride. Lo and Hi must always be >= 1<<16.

type RangeTable

```
type RangeTable struct {
    R16         []Range16
    R32         []Range32
    LatinOffset int // number of entries in R16 with Hi <= MaxLatin1
}
```

RangeTable defines a set of Unicode code points by listing the ranges of code points within the set. The ranges are listed in two slices to save space: a slice of 16-bit ranges and a slice of 32-bit ranges. The two slices must be in sorted order and non-overlapping. Also, R32 should contain only values >= 0x10000 (1<<16).

type SpecialCase

```
type SpecialCase []CaseRange
```

SpecialCase represents language-specific case mappings such as Turkish. Methods of SpecialCase customize (by overriding) the standard mappings.

▶ Example

```
var AzeriCase SpecialCase = _TurkishCase
```

```
var TurkishCase SpecialCase = _TurkishCase
```

func (SpecialCase) ToLower

```
func (special SpecialCase) ToLower(r rune) rune
```

ToLower maps the rune to lower case giving priority to the special mapping.

func (SpecialCase) ToTitle

```
func (special SpecialCase) ToTitle(r rune) rune
```

ToTitle maps the rune to title case giving priority to the special mapping.

func (SpecialCase) ToUpper

```
func (special SpecialCase) ToUpper(r rune) rune
```

ToUpper maps the rune to upper case giving priority to the special mapping.

Notes

Bugs

- There is no mechanism for full case folding, that is, for characters that involve multiple runes in the input or output.

Source Files

View all

casetables.go
digit.go
graphic.go
letter.go
tables.go

Directories

utf16

Package utf16 implements encoding and decoding of UTF-16 sequences.

utf8

Package utf8 implements functions and constants to support text encoded in UTF-8.