

-gstr

gstrGolang

```
import "github.com/gogf/gf/v2/text/gstr"
```

<https://pkg.go.dev/github.com/gogf/gf/v2/text/gstr>



<https://pkg.go.dev/github.com/gogf/gf/v2/text/gstr>

IsNumeric

- IsNumerics
-

```
IsNumeric(s string) bool
```

-

```
func ExampleIsNumeric() {
    fmt.Println(gstr.IsNumeric("88"))
    fmt.Println(gstr.IsNumeric("3.1415926"))
    fmt.Println(gstr.IsNumeric("abc"))
    // Output:
    // true
    // true
    // false
}
```

LenRune

- LenRuneunicode
-

```
LenRune(str string) int
```

-

```
func ExampleLenRune() {
    var (
        str    = `GoFrame`
        result = gstr.LenRune(str)
    )
    fmt.Println(result)

    // Output:
    // 9
}
```

Content Menu

-
- - IsNumeric
- - LenRune
- - Repeat
- - ToLower
 - ToUpper
 - UcFirst
 - LcFirst
 - UcWords
 - IsLetterLower
 - IsLetterUpper
- - Compare
 - Equal
- - Split
 - SplitAndTrim
 - Join
 - JoinAny
 - Explode
 - Implode
 - ChunkSplit
 - Fields
- - AddSlashes
 - StripSlashes
 - QuoteMeta
- - Count
 - Countl
 - CountWords
 - CountChars
- - SearchArray
 - InArray
 - PrefixArray
- - CaseCamel
 - CaseCamelLower
 - CaseSnake
 - CaseSnakeScreaming
 - CaseSnakeFirstUpper
 - CaseKebab
 - CaseKebabScreaming
 - CaseDelimited
 - CaseDelimitedScreaming
- - Contains
 - Containsl
 - ContainsAny
- - Chr
 - Ord
 - OctStr
 - Reverse
 - NumberFormat
 - Shuffle
 - HideStr
 - Nl2Br
 - WordWrap
- - IsSubDomain
- - Parse

Repeat

- RepeatInputMultiplier
-

```
Repeat(input string, multiplier int) string
```

-

```
func ExampleRepeat() {
    var (
        input      = `goframe`
        multiplier = 3
        result     = gstr.Repeat(input, multiplier)
    )
    fmt.Println(result)

    // Output:
    // goframe goframe goframe
}
```

ToLower

- ToLowerUnicode
-

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

-

```
func ExampleToLower() {
    var (
        s      = `GOFRAME`
        result = gstr.ToLower(s)
    )
    fmt.Println(result)

    // Output:
    // goframe
}
```

ToUpper

- ToUppersUnicode
-

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

-

- - Pos
 - PosRune
 - PosI
 - PosRuneI
 - PosR
 - PosRuneR
 - PosRI
 - PosRIRune
- - Replace
 - ReplaceI
 - ReplaceByArray
 - ReplaceByArray
 - ReplaceByMap
 - ReplaceByMap
- - Str
 - StrEx
 - StrTill
 - StrTillEx
 - SubStr
 - SubStrRune
 - StrLimit
 - StrLimitRune
 - SubStrFrom
 - SubStrFromEx
 - SubStrFromR
 - SubStrFromREx
- /
 - Trim
 - TrimStr
 - TrimLeft
 - TrimLeftStr
 - TrimRight
 - TrimRightStr
 - TrimAll
 - HasPrefix
 - HasSuffix
- - CompareVersion
 - CompareVersionGo
- - Levenshtein
 - SimilarText
 - Soundex

```
func ExampleToUpper() {
    var (
        s      = `goframe`
        result = gstr.ToUpper(s)
    )
    fmt.Println(result)

    // Output:
    // GOFRAME
}
```

UcFirst

- UcFirsts
-

```
UcFirst(s string) string
```

-

```
func ExampleUcFirst() {
    var (
        s      = `hello`
        result = gstr.UcFirst(s)
    )
    fmt.Println(result)

    // Output:
    // Hello
}
```

LcFirst

- LcFirsts
-

```
LcFirst(s string) string
```

-

```
func ExampleLcFirst() {
    var (
        str      = `Goframe`
        result = gstr.LcFirst(str)
    )
    fmt.Println(result)

    // Output:
    // goframe
}
```

UcWords

- UcWordsstr
-

```
UcWords(str string) string
```

-

```
func ExampleUcWords() {
    var (
        str    = `hello world`
        result = gstr.UcWords(str)
    )
    fmt.Println(result)

    // Output:
    // Hello World
}
```

IsLetterLower

- IsLetterLowerb
-

```
IsLetterLower(b byte) bool
```

-

```
func ExampleIsLetterLower() {
    fmt.Println(gstr.IsLetterLower('a'))
    fmt.Println(gstr.IsLetterLower('A'))

    // Output:
    // true
    // false
}
```

IsLetterUpper

- IsLetterUpperb
-

```
IsLetterUpper(b byte) bool
```

-

```
func ExampleIsLetterUpper() {
    fmt.Println(gstr.IsLetterUpper('A'))
    fmt.Println(gstr.IsLetterUpper('a'))

    // Output:
    // true
    // false
}
```

Compare

- Compare a == b0a < b-1a > b+1
-

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

-

```
func ExampleCompare() {
    fmt.Println(gstr.Compare("c", "c"))
    fmt.Println(gstr.Compare("a", "b"))
    fmt.Println(gstr.Compare("c", "b"))

    // Output:
    // 0
    // -1
    // 1
}
```

Equal

- Equalab
-

```
Equal(a, b string) bool
```

-

```
func ExampleEqual() {
    fmt.Println(gstr.Equal(`A`, `a`))
    fmt.Println(gstr.Equal(`A`, `A`))
    fmt.Println(gstr.Equal(`A`, `B`))

    // Output:
    // true
    // true
    // false
}
```

Split

- Splitdelimiterstr[]string
-

```
Split(str, delimiter string) []string
```

-

```
func ExampleSplit() {
    var (
        str      = `a|b|c|d`
        delimiter = `|`
        result    = gstr.Split(str, delimiter)
    )
    fmt.Printf(`%#v`, result)

    // Output:
    // []string{"a", "b", "c", "d"}
}
```

SplitAndTrim

- SplitAndTrimdelimiterstr[]string[]stringTrimTrim
-

```
SplitAndTrim(str, delimiter string, characterMask ...string) []string
```

-

```
func ExampleSplitAndTrim() {
    var (
        str      = `a|b||||c|d`
        delimiter = `|`
        result    = gstr.SplitAndTrim(str, delimiter)
    )
    fmt.Printf(`%#v`, result)

    // Output:
    // []string{"a", "b", "c", "d"}
}
```

Join

- Joinarraysep

-

```
Join(array []string, sep string) string
```

-

```
func ExampleJoin() {
    var (
        array = []string{"goframe", "is", "very", "easy",
"to", "use"}
        sep    = ` `
        result = gstr.Join(array, sep)
    )
    fmt.Println(result)

    // Output:
    // goframe is very easy to use
}
```

JoinAny

- JoinAnyarrayseparray

-

```
JoinAny(array interface{}, sep string) string
```

-

```
func ExampleJoinAny() {
    var (
        sep    = ` , `
        arr2    = []int{99, 73, 85, 66}
        result = gstr.JoinAny(arr2, sep)
    )
    fmt.Println(result)

    // Output:
    // 99,73,85,66
}
```

Explode

- Explodedelimiterstr[]string
-

```
Explode(delimiter, str string) []string
```

-

```
func ExampleExplode() {
    var (
        str      = `Hello World`
        delimiter = " "
        result    = gstr.Explode(delimiter, str)
    )
    fmt.Printf(`%#v`, result)

    // Output:
    // []string{"Hello", "World"}
}
```

Implode

- Implodegluepieces
-

```
Implode(glue string, pieces []string) string
```

-

```
func ExampleImplode() {
    var (
        pieces = []string{"goframe", "is", "very", "easy",
        "to", "use"}
        glue    = " "
        result  = gstr.Implode(glue, pieces)
    )
    fmt.Println(result)

    // Output:
    // goframe is very easy to use
}
```

ChunkSplit

- ChunkSplitchunkLenend
-

```
ChunkSplit(body string, chunkLen int, end string) string
```

-

```
func ExampleChunkSplit() {
    var (
        body      = `1234567890`
        chunkLen = 2
        end        = "#"
        result     = gstr.ChunkSplit(body, chunkLen, end)
    )
    fmt.Println(result)

    // Output:
    // 12#34#56#78#90#
}
```

Fields

- Fields[]string
-

```
Fields(str string) []string
```

-

```
func ExampleFields() {
    var (
        str      = `Hello World`
        result = gstr.Fields(str)
    )
    fmt.Printf(`%#v`, result)

    // Output:
    // []string{"Hello", "World"}
}
```

AddSlashes

- AddSlashes`\'`
-

```
AddSlashes(str string) string
```

-

```
func ExampleAddSlashes() {
    var (
        str      = `\'aa\'\'bb\'cc\'r\'n\'d\'t\'`
        result = gstr.AddSlashes(str)
    )

    fmt.Println(result)

    // Output:
    // \'aa\'\'\'bb\'\'cc\'\'r\'\'n\'\'d\'\'t
}
```

StripSlashes

- StripSlashesstr'\'
-

```
StripSlashes(str string) string
```

-

```
func ExampleStripSlashes() {
    var (
        str      = `C:\\windows\\GoFrame\\test`
        result = gstr.StripSlashes(str)
    )
    fmt.Println(result)

    // Output:
    // C:\windows\GoFrame\test
}
```

QuoteMeta

- QuoteMetastr'. \ + * ? [^] (\$) '\'
-

```
QuoteMeta(str string, chars ...string) string
```

-

```
func ExampleQuoteMeta() {
    {
        var (
            str      = `.\.+?[^]()`
            result = gstr.QuoteMeta(str)
        )
        fmt.Println(result)
    }
    {
        var (
            str      = `https://goframe.org/pages/viewpage.
action?pageId=1114327`
            result = gstr.QuoteMeta(str)
        )
        fmt.Println(result)
    }

    // Output:
    // \.\\.\\+\\?\\[\\^\\]\\(\\)
    // https://goframe\\.org/pages/viewpage\\.action\\?
    // pageId=1114327
}
```

Count

- Countsubstrs substr0
-

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

-

```
func ExampleCount() {
    var (
        str      = `goframe is very, very easy to use`
        substr1 = "goframe"
        substr2 = "very"
        result1 = gstr.Count(str, substr1)
        result2 = gstr.Count(str, substr2)
    )
    fmt.Println(result1)
    fmt.Println(result2)

    // Output:
    // 1
    // 2
}
```

CountI

- Countsubstrings substr0
-

```
CountI(s, substr string) int
```

-

```
func ExampleCountI() {
    var (
        str      = `goframe is very, very easy to use`
        substr1 = "GOFRAME"
        substr2 = "VERY"
        result1 = gstr.CountI(str, substr1)
        result2 = gstr.CountI(str, substr2)
    )
    fmt.Println(result1)
    fmt.Println(result2)

    // Output:
    // 1
    // 2
}
```

CountWords

- CountWordsmap[string]intstr
-

```
CountWords(str string) map[string]int
```

-

```
func ExampleCountWords() {
    var (
        str      = `goframe is very, very easy to use!`
        result = gstr.CountWords(str)
    )
    fmt.Printf(`%#v`, result)

    // Output:
    // map[string]int{"easy":1, "goframe":1, "is":1, "to":1,
    "use!":1, "very":1, "very,":1}
}
```

CountChars

- CountCharsmap[string]intstrnoSpace
-

```
CountChars(str string, noSpace ...bool) map[string]int
```

-

```
func ExampleCountChars() {
    var (
        str      = `goframe`
        result = gstr.CountChars(str)
    )
    fmt.Println(result)

    // May Output:
    // map[a:1 e:1 f:1 g:1 m:1 o:1 r:1]
}
```

SearchArray

- SearchArray[]string 'a''s''a' 'a''s'-1
-

```
SearchArray(a []string, s string) int
```

-

```
func ExampleSearchArray() {
    var (
        array = []string{"goframe", "is", "very", "nice"}
        str    = `goframe`
        result = gstr.SearchArray(array, str)
    )
    fmt.Println(result)

    // Output:
    // 0
}
```

InArray

- InArray[]string 'a' s '

-

```
InArray(a []string, s string) bool
```

-

```
func ExampleInArray() {  
    var (  
        a      = []string{"goframe", "is", "very", "easy",  
"to", "use"}  
        s      = "goframe"  
        result = gstr.InArray(a, s)  
    )  
    fmt.Println(result)  
  
    // Output:  
    // true  
}
```

PrefixArray

- PrefixArray([]string array'prefix')
-

```
PrefixArray(array []string, prefix string)
```

-

```
func ExamplePrefixArray() {  
    var (  
        strArray = []string{"tom", "lily", "john"}  
    )  
  
    gstr.PrefixArray(strArray, "classA_")  
  
    fmt.Println(strArray)  
  
    // Output:  
    // [classA_tom classA_lily classA_john]  
}
```

CaseCamel

- CaseCamel()
-

```
CaseCamel(s string) string
```

-

```
func ExampleCaseCamel() {
    var (
        str    = `hello world`
        result = gstr.CaseCamel(str)
    )
    fmt.Println(result)

    // Output:
    // HelloWorld
}
```

CaseCamelLower

- CaseCamelLower()
-

```
CaseCamelLower(s string) string
```

-

```
func ExampleCaseCamelLower() {
    var (
        str    = `hello world`
        result = gstr.CaseCamelLower(str)
    )
    fmt.Println(result)

    // Output:
    // helloWorld
}
```

CaseSnake

- CaseSnake(...)(_),
-

```
CaseSnake(s string) string
```

-

```
func ExampleCaseSnake() {
    var (
        str    = `hello world`
        result = gstr.CaseSnake(str)
    )
    fmt.Println(result)

    // Output:
    // hello_world
}
```

CaseSnakeScreaming

- CaseSnakeScreaming(...),'_','',
-

```
CaseSnakeScreaming(s string) string
```

-

```
func ExampleCaseSnakeScreaming() {
    var (
        str    = `hello world`
        result = gstr.CaseSnakeScreaming(str)
    )
    fmt.Println(result)

    // Output:
    // HELLO_WORLD
}
```

CaseSnakeFirstUpper

- CaseSnakeFirstUpper,'_','_', '_'
-

```
CaseSnakeFirstUpper(word string, underscore ...string) string
```

-

```
func ExampleCaseSnakeFirstUpper() {
    var (
        str    = `RGBCodeMd5`
        result = gstr.CaseSnakeFirstUpper(str)
    )
    fmt.Println(result)

    // Output:
    // rgb_code_md5
}
```

CaseKebab

- CaseKebab(...)'-' ,
-

```
CaseKebab(s string) string
```

-

```
func ExampleCaseKebab() {
    var (
        str    = `hello world`
        result = gstr.CaseKebab(str)
    )
    fmt.Println(result)

    // Output:
    // hello-world
}
```

CaseKebabScreaming

- CaseKebabScreaming(...)'-' ,
-

```
CaseKebabScreaming(s string) string
```

-

```
func ExampleCaseKebabScreaming() {
    var (
        str    = `hello world`
        result = gstr.CaseKebabScreaming(str)
    )
    fmt.Println(result)

    // Output:
    // HELLO-WORLD
}
```

CaseDelimited

- CaseDelimited
-

```
CaseDelimited(s string, del byte) string
```

-

```
func ExampleCaseDelimited() {
    var (
        str    = `hello world`
        del    = byte('-')
        result = gstr.CaseDelimited(str, del)
    )
    fmt.Println(result)

    // Output:
    // hello-world
}
```

CaseDelimitedScreaming

- CaseDelimitedScreaming(,,,),true,false
-

```
CaseDelimitedScreaming(s string, del uint8, screaming bool) string
```

-

```

func ExampleCaseDelimitedScreaming() {
    {
        var (
            str    = `hello world`
            del    = byte('-')
            result = gstr.CaseDelimitedScreaming(str,
del, true)
        )
        fmt.Println(result)
    }
    {
        var (
            str    = `hello world`
            del    = byte('-')
            result = gstr.CaseDelimitedScreaming(str,
del, false)
        )
        fmt.Println(result)
    }

    // Output:
    // HELLO-WORLD
    // hello-world
}

```

Contains

- Containsstrsubstr

```
Contains(str, substr string) bool
```

-

```

func ExampleContains() {
    {
        var (
            str    = `Hello World`
            substr = `Hello`
            result = gstr.Contains(str, substr)
        )
        fmt.Println(result)
    }
    {
        var (
            str    = `Hello World`
            substr = `hello`
            result = gstr.Contains(str, substr)
        )
        fmt.Println(result)
    }

    // Output:
    // true
    // false
}

```

ContainsI

- ContainsIsubstrstr

-

```
ContainsI(str, substr string) bool
```

-

```
func ExampleContainsI() {
    var (
        str      = `Hello World`
        substr    = "hello"
        result1 = gstr.Contains(str, substr)
        result2 = gstr.ContainsI(str, substr)
    )
    fmt.Println(result1)
    fmt.Println(result2)

    // Output:
    // false
    // true
}
```

ContainsAny

- ContainsAnyschars

-

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

-

```
func ExampleContainsAny() {
    {
        var (
            s      = `goframe`
            chars = "g"
            result = gstr.ContainsAny(s, chars)
        )
        fmt.Println(result)
    }
    {
        var (
            s      = `goframe`
            chars = "G"
            result = gstr.ContainsAny(s, chars)
        )
        fmt.Println(result)
    }

    // Output:
    // true
    // false
}
```

Chr

- Chr0-255ascii

-

```
Chr(ascii int) string
```

-

```
func ExampleChr() {  
    var (  
        ascii = 65 // A  
        result = gstr.Chr(ascii)  
    )  
    fmt.Println(result)  
  
    // Output:  
    // A  
}
```

Ord

- Ord0-255

-

```
Ord(char string) int
```

-

```
func ExampleOrd() {  
    var (  
        str    = `goframe`  
        result = gstr.Ordinal(str)  
    )  
  
    fmt.Println(result)  
  
    // Output:  
    // 103  
}
```

OctStr

- OctStrstr

-

```
OctStr(str string) string
```

-

```
func ExampleOctStr() {  
    var (  
        str    = `346200241`  
        result = gstr.OctStr(str)  
    )  
    fmt.Println(result)  
  
    // Output:  
    //  
}
```

Reverse

- Reversestr

-

```
Reverse(str string) string
```

-

```
func ExampleReverse() {  
    var (  
        str    = `123456`  
        result = gstr.Reverse(str)  
    )  
    fmt.Println(result)  
  
    // Output:  
    // 654321  
}
```

NumberFormat

- NumberFormat
 - decimal
 - decPoint
 - thousand
-

```
NumberFormat(number float64, decimals int, decPoint, thousandsSep  
string) string
```

-

```
func ExampleNumberFormat() {  
    var (  
        number      float64 = 123456  
        decimals     = 2  
        decPoint     = "."  
        thousandsSep = ","  
        result       = gstr.NumberFormat(number,  
decimals, decPoint, thousandsSep)  
    )  
    fmt.Println(result)  
  
    // Output:  
    // 123,456.00  
}
```

Shuffle

- Shufflestr
-

```
Shuffle(str string) string
```

-

```
func ExampleShuffle() {
    var (
        str    = `123456`
        result = gstr.Shuffle(str)
    )
    fmt.Println(result)

    // May Output:
    // 563214
}
```

HideStr

- HideStr(str string, percent int, hide string) string
-

```
HideStr(str string, percent int, hide string) string
```

-

```
func ExampleHideStr() {
    var (
        str      = `13800138000`
        percent = 40
        hide     = `*`
        result  = gstr.HideStr(str, percent, hide)
    )
    fmt.Println(result)

    // Output:
    // 138***8000
}
```

Nl2Br

- Nl2BrHTML(' br ' |
): \n\r \r\n \r \n
-

```
Nl2Br(str string, isXhtml ...bool) string
```

-

```
func ExampleNl2Br() {
    var (
        str = `goframe
is
very
easy
to
use`
    )
    result = gstr.Nl2Br(str)

    fmt.Println(result)

    // Output:
    // goframe<br>is<br>very<br>easy<br>to<br>use
}
```

WordWrap

- WordWrapstr
-

```
WordWrap(str string, width int, br string) string
```

-

```
func ExampleWordWrap() {
    {
        var (
            str      = `A very long wooooooooooooooooooooord.
and something`
            width   = 8
            br      = "\n"
            result  = gstr.WordWrap(str, width, br)
        )
        fmt.Println(result)
    }
    {
        var (
            str      = `The quick brown fox jumped over
the lazy dog.`
            width   = 20
            br      = "<br />\n"
            result  = gstr.WordWrap(str, width, br)
        )
        fmt.Printf("%v", result)
    }

    // Output:
    // A very
    // long
    // wooooooooooooooooooooord.
    // and
    // something
    // The quick brown fox<br />
    // jumped over the lazy<br />
    // dog.
}
```

IsSubDomain

- IsSubDomainsubDomainmainDomain mainDomain''
-

```
IsSubDomain(subDomain string, mainDomain string) bool
```

-

```
func ExampleIsSubDomain() {  
    var (  
        subDomain = `s.goframe.org`  
        mainDomain = `goframe.org`  
        result     = gstr.IsSubDomain(subDomain, mainDomain)  
    )  
    fmt.Println(result)  
  
    // Output:  
    // true  
}
```

Parse

- Parsemap[string]interface{}
-

```
Parse(s string) (result map[string]interface{}, err error)
```

-

```

func ExampleParse() {
    {
        var (
            str      = `v1=m&v2=n`
            result, _ = gstr.Parse(str)
        )
        fmt.Println(result)
    }
    {
        var (
            str      = `v[a][a]=m&v[a][b]=n`
            result, _ = gstr.Parse(str)
        )
        fmt.Println(result)
    }
    {
        // The form of nested Slice is not yet supported.
        var str = `v[][]=m&v[][]=n`
        result, err := gstr.Parse(str)
        if err != nil {
            panic(err)
        }
        fmt.Println(result)
    }
    {
        // This will produce an error.
        var str = `v=m&v[a]=n`
        result, err := gstr.Parse(str)
        if err != nil {
            println(err)
        }
        fmt.Println(result)
    }
    {
        var (
            str      = `a .[[b=c`
            result, _ = gstr.Parse(str)
        )
        fmt.Println(result)
    }

    // May Output:
    // map[v1:m v2:n]
    // map[v:map[a:map[a:m b:n]]]
    // map[v:map[]]
    // Error: expected type 'map[string]interface{}' for key
    'v', but got 'string'
    // map[]
    // map[a____[b:c]
}

```

Pos

- Posneedlehaystack -1

```
Pos(haystack, needle string, startOffset ...int) int
```

-

```
func ExamplePos() {
    var (
        haystack = `Hello World`
        needle    = `World`
        result    = gstr.Pos(haystack, needle)
    )
    fmt.Println(result)

    // Output:
    // 6
}
```

PosRune

- PosRunePoshaystackneedleunicode
-

```
PosRune(haystack, needle string, startOffset ...int) int
```

-

```
func ExamplePosRune() {
    var (
        haystack = `GoFrameGo`
        needle    = `Go`
        posI      = gstr.PosRune(haystack, needle)
        posR      = gstr.PosRRune(haystack, needle)
    )
    fmt.Println(posI)
    fmt.Println(posR)

    // Output:
    // 0
    // 22
}
```

PosI

- PosIneedlehaystack -1
-

```
PosI(haystack, needle string, startOffset ...int) int
```

-

```
func ExamplePosI() {
    var (
        haystack = `goframe is very, very easy to use`
        needle    = `very`
        posI      = gstr.PosI(haystack, needle)
        posR      = gstr.PosR(haystack, needle)
    )
    fmt.Println(posI)
    fmt.Println(posR)

    // Output:
    // 11
    // 17
}
```

PosRuneI

- PosRuneIPosIhaystackneedleunicode
-

```
PosIRune(haystack, needle string, startOffset ...int) int
```

-

```
func ExamplePosIRune() {
    {
        var (
            haystack = `GoFrameGo`
            needle    = ``
            startOffset = 10
            result    = gstr.PosIRune(haystack,
needle, startOffset)
        )
        fmt.Println(result)
    }
    {
        var (
            haystack = `GoFrameGo`
            needle    = ``
            startOffset = 30
            result    = gstr.PosIRune(haystack,
needle, startOffset)
        )
        fmt.Println(result)
    }

    // Output:
    // 14
    // -1
}
```

PosR

- PosRneedlehaystack -1
-

```
PosR(haystack, needle string, startOffset ...int) int
```

-

```
func ExamplePosR() {
    var (
        haystack = `goframe is very, very easy to use`
        needle    = `very`
        posI      = gstr.PosI(haystack, needle)
        posR      = gstr.PosR(haystack, needle)
    )
    fmt.Println(posI)
    fmt.Println(posR)

    // Output:
    // 11
    // 17
}
```

PosRuneR

- PosRuneRPosRhaystackneedleunicode
-

```
PosRRune(haystack, needle string, startOffset ...int) int
```

-

```
func ExamplePosRRune() {
    var (
        haystack = `GoFrameGo`
        needle    = `Go`
        posI      = gstr.PosIRune(haystack, needle)
        posR      = gstr.PosRRune(haystack, needle)
    )
    fmt.Println(posI)
    fmt.Println(posR)

    // Output:
    // 0
    // 22
}
```

PosRI

- PosRIneedlehaystack -1
-

```
PosRI(haystack, needle string, startOffset ...int) int
```

-

```
func ExamplePosRI() {
    var (
        haystack = `goframe is very, very easy to use`
        needle    = `VERY`
        posI      = gstr.PosI(haystack, needle)
        posR      = gstr.PosRI(haystack, needle)
    )
    fmt.Println(posI)
    fmt.Println(posR)

    // Output:
    // 11
    // 17
}
```

PosRIRune

- PosRIRunePosRIhaystackneedleunicode
-

```
PosRIRune(haystack, needle string, startOffset ...int) int
```

-

```
func ExamplePosRIRune() {
    var (
        haystack = `GoFrameGo`
        needle    = `GO`
        posI      = gstr.PosIRune(haystack, needle)
        posR      = gstr.PosRIRune(haystack, needle)
    )
    fmt.Println(posI)
    fmt.Println(posR)

    // Output:
    // 0
    // 22
}
```

Replace

- Replaceorigin, searchreplacesearch
-

```
Replace(origin, search, replace string, count ...int) string
```

-

```
func ExampleReplace() {
    var (
        origin = `golang is very nice!`
        search  = `golang`
        replace = `goframe`
        result  = gstr.Replace(origin, search, replace)
    )
    fmt.Println(result)

    // Output:
    // goframe is very nice!
}
```

ReplaceI

- ReplaceIorigin, searchreplacesearch
-

```
ReplaceI(origin, search, replace string, count ...int) string
```

-

```

func ExampleReplaceI() {
    var (
        origin = `golang is very nice!`
        search  = `GOLANG`
        replace = `goframe`
        result  = gstr.ReplaceI(origin, search, replace)
    )
    fmt.Println(result)

    // Output:
    // goframe is very nice!
}

```

ReplaceByArray

- `ReplaceByArrayorigin(search, replace)`
-

```
ReplaceByArray(origin string, array []string) string
```

-

```

func ExampleReplaceByArray() {
    {
        var (
            origin = `golang is very nice`
            array  = []string{"lang", "frame"}
            result = gstr.ReplaceByArray(origin, array)
        )
        fmt.Println(result)
    }
    {
        var (
            origin = `golang is very good`
            array  = []string{"golang", "goframe",
"good", "nice"}
            result = gstr.ReplaceByArray(origin, array)
        )
        fmt.Println(result)
    }

    // Output:
    // goframe is very nice
    // goframe is very nice
}

```

ReplaceIByArray

- `ReplaceIByArrayorigin(search, replace)`
-

```
ReplaceIByArray(origin string, array []string) string
```

-

```
func ExampleReplaceIByArray() {
    var (
        origin = `golang is very Good`
        array = []string{"Golang", "goframe", "GOOD",
"nice"}
        result = gstr.ReplaceIByArray(origin, array)
    )

    fmt.Println(result)

    // Output:
    // goframe is very nice
}
```

ReplaceByMap

- ReplaceByMaporiginmapkeyvalue
-

```
ReplaceByMap(origin string, replaces map[string]string) string
```

-

```
func ExampleReplaceByMap() {
    {
        var (
            origin = `golang is very nice`
            replaces = map[string]string{
                "lang": "frame",
            }
            result = gstr.ReplaceByMap(origin, replaces)
        )
        fmt.Println(result)
    }
    {
        var (
            origin = `golang is very good`
            replaces = map[string]string{
                "golang": "goframe",
                "good": "nice",
            }
            result = gstr.ReplaceByMap(origin, replaces)
        )
        fmt.Println(result)
    }

    // Output:
    // goframe is very nice
    // goframe is very nice
}
```

ReplaceIByMap

- ReplaceIByMaporiginmapkeyvalue
-

```
ReplaceIByMap(origin string, replaces map[string]string) string
```

-

```

func ExampleReplaceIByMap() {
    var (
        origin   = `golang is very nice`
        replaces = map[string]string{
            "Lang": "frame",
        }
        result = gstr.ReplaceIByMap(origin, replaces)
    )
    fmt.Println(result)

    // Output:
    // goframe is very nice
}

```

Str

- Strneedlehaystackneedle
-

```
Str(haystack string, needle string) string
```

-

```

func ExampleStr() {
    var (
        haystack = `xxx.jpg`
        needle    = `.`
        result    = gstr.Str(haystack, needle)
    )
    fmt.Println(result)

    // Output:
    // .jpg
}

```

StrEx

- StrExneedlehaystackneedle
-

```
StrEx(haystack string, needle string) string
```

-

```

func ExampleStrEx() {
    var (
        haystack = `https://goframe.org/index.html?a=1&b=2`
        needle    = `?`
        result    = gstr.StrEx(haystack, needle)
    )
    fmt.Println(result)

    // Output:
    // a=1&b=2
}

```

StrTill

- StrTillhaystackneedleneedle
-

```
StrTill(haystack string, needle string) string
```

-

```
func ExampleStrTill() {
    var (
        haystack = `https://goframe.org/index.html?
test=123456`
        needle   = `?`
        result   = gstr.StrTill(haystack, needle)
    )
    fmt.Println(result)

    // Output:
    // https://goframe.org/index.html?
}
```

StrTillEx

- StrTillExhaystackneedleneedle
-

```
StrTillEx(haystack string, needle string) string
```

-

```
func ExampleStrTillEx() {
    var (
        haystack = `https://goframe.org/index.html?
test=123456`
        needle   = `?`
        result   = gstr.StrTillEx(haystack, needle)
    )
    fmt.Println(result)

    // Output:
    // https://goframe.org/index.html
}
```

SubStr

- SubStrstrstartlength lengthstr
-

```
SubStr(str string, start int, length ...int) (substr string)
```

-

```
func ExampleSubStr() {
    var (
        str    = `1234567890`
        start  = 0
        length = 4
        subStr = gstr.SubStr(str, start, length)
    )
    fmt.Println(subStr)

    // Output:
    // 1234
}
```

SubStrRune

- SubStrRuneunicodestrstartlength lengthstr
-

```
SubStrRune(str string, start int, length ...int) (substr string)
```

-

```
func ExampleSubStrRune() {
    var (
        str    = `GoFrameGo`
        start  = 14
        length = 3
        subStr = gstr.SubStrRune(str, start, length)
    )
    fmt.Println(subStr)

    // Output:
    //
}
```

StrLimit

- StrLimitstrlengthsuffix...
-

```
StrLimit(str string, length int, suffix ...string) string
```

-

```
func ExampleStrLimit() {
    var (
        str    = `123456789`
        length = 3
        suffix = `...`
        result = gstr.StrLimit(str, length, suffix)
    )
    fmt.Println(result)

    // Output:
    // 123...
}
```

StrLimitRune

- StrLimitRuneunicodestrlengthsuffix...

-

```
StrLimitRune(str string, length int, suffix ...string) string
```

-

```
func ExampleStrLimitRune() {
    var (
        str    = `GoFrameGo`
        length = 17
        suffix = "... "
        result = gstr.StrLimitRune(str, length, suffix)
    )
    fmt.Println(result)

    // Output:
    // GoFrame...
}
```

SubStrFrom

- SubStrFromstrneedstrneed
-

```
SubStrFrom(str string, need string) (substr string)
```

-

```
func ExampleSubStrFrom() {
    var (
        str = "GoFrameGood"
        need = ``
    )

    fmt.Println(gstr.SubStrFrom(str, need))

    // Output:
    // GoFrameGood
}
```

SubStrFromEx

- SubStrFromExstrneedstrneed
-

```
SubStrFromEx(str string, need string) (substr string)
```

-

```
func ExampleSubStrFromEx() {
    var (
        str = "GoFrameGood"
        need = ``
    )

    fmt.Println(gstr.SubStrFromEx(str, need))

    // Output:
    // GoFrameGood
}
```

SubStrFromR

- SubStrFromRstrneedstrneed
-

```
SubStrFromR(str string, need string) (substr string)
```

-

```
func ExampleSubStrFromR() {  
    var (  
        str = "GoFrameGood"  
        need = `Go`  
    )  
  
    fmt.Println(gstr.SubStrFromR(str, need))  
  
    // Output:  
    // Good  
}
```

SubStrFromREx

- SubStrFromRExstrneedstrneed
-

```
SubStrFromREx(str string, need string) (substr string)
```

-

```
func ExampleSubStrFromREx() {  
    var (  
        str = "GoFrameGood"  
        need = `Go`  
    )  
  
    fmt.Println(gstr.SubStrFromREx(str, need))  
  
    // Output:  
    // od  
}
```

/

Trim

- Trim() characterMask
-

```
Trim(str string, characterMask ...string) string
```

-

```
func ExampleTrim() {
    var (
        str           = `*Hello World*`
        characterMask = "*"
        result         = gstr.Trim(str, characterMask)
    )
    fmt.Println(result)

    // Output:
    // Hello World
}
```

TrimStr

- TrimStrcut
-

```
TrimStr(str string, cut string, count ...int) string
```

-

```
func ExampleTrimStr() {
    var (
        str    = `Hello World`
        cut    = "World"
        count  = -1
        result = gstr.TrimStr(str, cut, count)
    )
    fmt.Println(result)

    // Output:
    // Hello
}
```

TrimLeft

- TrimLeft()
-

```
TrimLeft(str string, characterMask ...string) string
```

-

```
func ExampleTrimLeft() {
    var (
        str           = `*Hello World*`
        characterMask = "*"
        result         = gstr.TrimLeft(str, characterMask)
    )
    fmt.Println(result)

    // Output:
    // Hello World*
}
```

TrimLeftStr

- TrimLeftStrcountcut
-

```
TrimLeftStr(str string, cut string, count ...int) string
```

-

```
func ExampleTrimLeftStr() {  
    var (  
        str      = `**Hello World**`  
        cut      = ""  
        count    = 1  
        result   = gstr.TrimLeftStr(str, cut, count)  
    )  
    fmt.Println(result)  
  
    // Output:  
    // **Hello World**  
}
```

TrimRight

- TrimRight()

-

```
TrimRight(str string, characterMask ...string) string
```

-

```
func ExampleTrimRight() {  
    var (  
        str              = `**Hello World**`  
        characterMask    = "**def" // []byte{"*", "d", "e", "f"}  
        result           = gstr.TrimRight(str, characterMask)  
    )  
    fmt.Println(result)  
  
    // Output:  
    // **Hello Worl
```

TrimRightStr

- TrimRightStrcountcut

-

```
TrimRightStr(str string, cut string, count ...int) string
```

-

```
func ExampleTrimRightStr() {  
    var (  
        str    = `Hello World!`  
        cut    = "!"  
        count  = -1  
        result = gstr.TrimRightStr(str, cut, count)  
    )  
    fmt.Println(result)  
  
    // Output:  
    // Hello World  
}
```

TrimAll

- TrimAllstr()characterMask
-

```
TrimAll(str string, characterMask ...string) string
```

-

```
func ExampleTrimAll() {
    var (
        str           = `*Hello World*`
        characterMask = "*"
        result        = gstr.TrimAll(str, characterMask)
    )
    fmt.Println(result)

    // Output:
    // HelloWorld
}
```

HasPrefix

- HasPrefixsprefix
-

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

-

```
func ExampleHasPrefix() {
    var (
        s       = `Hello World`
        prefix = "Hello"
        result = gstr.HasPrefix(s, prefix)
    )
    fmt.Println(result)

    // Output:
    // true
}
```

HasSuffix

- HasSuffixssuffix
-

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

-

```
func ExampleHasSuffix() {
    var (
        s      = `my best love is goframe`
        prefix = "goframe"
        result = gstr.HasSuffix(s, prefix)
    )
    fmt.Println(result)

    // Output:
    // true
}
```

CompareVersion

- CompareVersionabGNU

```
CompareVersion(a, b string) int
```

-

```
func ExampleCompareVersion() {
    fmt.Println(gstr.CompareVersion("v2.11.9", "v2.10.8"))
    fmt.Println(gstr.CompareVersion("1.10.8", "1.19.7"))
    fmt.Println(gstr.CompareVersion("2.8.beta", "2.8"))

    // Output:
    // 1
    // -1
    // 0
}
```

CompareVersionGo

- CompareVersionGoabGolang

```
CompareVersionGo(a, b string) int
```

-

```
func ExampleCompareVersionGo() {
    fmt.Println(gstr.CompareVersionGo("v2.11.9", "v2.10.8"))
    fmt.Println(gstr.CompareVersionGo("v4.20.1", "v4.20.1
+incompatible"))
    fmt.Println(gstr.CompareVersionGo(
        "v0.0.2-20180626092158-b2ccc119800e",
        "v1.0.1-20190626092158-b2ccc519800e",
    ))

    // Output:
    // 1
    // 1
    // -1
}
```

Levenshtein

- LevenshteinLevenshtein
-

```
Levenshtein(str1, str2 string, costIns, costRep, costDel int) int
```

-

```
func ExampleLevenshtein() {
    var (
        str1    = "Hello World"
        str2    = "hallo World"
        costIns = 1
        costRep = 1
        costDel = 1
        result  = gstr.Levenshtein(str1, str2, costIns,
costRep, costDel)
    )
    fmt.Println(result)

    // Output:
    // 2
}
```

SimilarText

- SimilarText
-

```
SimilarText(first, second string, percent *float64) int
```

-

```
func ExampleSimilarText() {
    var (
        first  = `AaBbCcDd`
        second = `ad`
        percent = 0.80
        result = gstr.SimilarText(first, second, &percent)
    )
    fmt.Println(result)

    // Output:
    // 2
}
```

Soundex

- SoundexSoundex
-

```
Soundex(str string) string
```

-

```
func ExampleSoundex() {  
    var (  
        str1    = `Hello`  
        str2    = `Hallo`  
        result1 = gstr.Soundex(str1)  
        result2 = gstr.Soundex(str2)  
    )  
    fmt.Println(result1, result2)  
  
    // Output:  
    // H400 H400  
}
```