Golang strconv库

# AppendXXX 将元素经过类型转换的处理以后追加到数组中

## [func AppendBool(dst []byte, b bool) []byte](https://pkg.go.dev/strconv@go1.19.4" \l "AppendBool)

## [func AppendFloat(dst []byte, f float64, fmt byte, prec, bitSize int) []byte](https://pkg.go.dev/strconv@go1.19.4" \l "AppendFloat)

## [func AppendInt(dst []byte, i int64, base int) []byte](https://pkg.go.dev/strconv@go1.19.4" \l "AppendInt)

## [func AppendQuote(dst []byte, s string) []byte](https://pkg.go.dev/strconv@go1.19.4" \l "AppendQuote)

## [func AppendQuoteRune(dst []byte, r rune) []byte](https://pkg.go.dev/strconv@go1.19.4" \l "AppendQuoteRune)

## [func AppendQuoteRuneToASCII(dst []byte, r rune) []byte](https://pkg.go.dev/strconv@go1.19.4" \l "AppendQuoteRuneToASCII)

## [func AppendQuoteRuneToGraphic(dst []byte, r rune) []byte](https://pkg.go.dev/strconv@go1.19.4" \l "AppendQuoteRuneToGraphic)

## [func AppendQuoteToASCII(dst []byte, s string) []byte](https://pkg.go.dev/strconv@go1.19.4" \l "AppendQuoteToASCII)

## [func AppendQuoteToGraphic(dst []byte, s string) []byte](https://pkg.go.dev/strconv@go1.19.4" \l "AppendQuoteToGraphic)

## [func AppendUint(dst []byte, i uint64, base int) []byte](https://pkg.go.dev/strconv@go1.19.4" \l "AppendUint)

## [func Atoi(s string) (int, error)](https://pkg.go.dev/strconv@go1.19.4" \l "Atoi)：string类型转int

## [func CanBackquote(s string) bool](https://pkg.go.dev/strconv@go1.19.4" \l "CanBackquote)

# 各个类型转string

## [func FormatBool(b bool) string](https://pkg.go.dev/strconv@go1.19.4" \l "FormatBool)：bool类型转string

## [func FormatComplex(c complex128, fmt byte, prec, bitSize int) string](https://pkg.go.dev/strconv@go1.19.4" \l "FormatComplex)：complex类型转string

## [func FormatFloat(f float64, fmt byte, prec, bitSize int) string](https://pkg.go.dev/strconv@go1.19.4" \l "FormatFloat)：float64转string

## [func FormatInt(i int64, base int) string](https://pkg.go.dev/strconv@go1.19.4" \l "FormatInt)

## [func FormatUint(i uint64, base int) string](https://pkg.go.dev/strconv@go1.19.4" \l "FormatUint)

## [func IsGraphic(r rune) bool](https://pkg.go.dev/strconv@go1.19.4" \l "IsGraphic)

## [func IsPrint(r rune) bool](https://pkg.go.dev/strconv@go1.19.4" \l "IsPrint)

## [func Itoa(i int) string](https://pkg.go.dev/strconv@go1.19.4" \l "Itoa)

String转各个类型

## [func ParseBool(str string) (bool, error)](https://pkg.go.dev/strconv@go1.19.4" \l "ParseBool)

## [func ParseComplex(s string, bitSize int) (complex128, error)](https://pkg.go.dev/strconv@go1.19.4" \l "ParseComplex)

## [func ParseFloat(s string, bitSize int) (float64, error)](https://pkg.go.dev/strconv@go1.19.4" \l "ParseFloat)

## [func ParseInt(s string, base int, bitSize int) (i int64, err error)](https://pkg.go.dev/strconv@go1.19.4" \l "ParseInt)

## [func ParseUint(s string, base int, bitSize int) (uint64, error)](https://pkg.go.dev/strconv@go1.19.4" \l "ParseUint)

## [func Quote(s string) string](https://pkg.go.dev/strconv@go1.19.4" \l "Quote)

## [func QuoteRune(r rune) string](https://pkg.go.dev/strconv@go1.19.4" \l "QuoteRune)

## [func QuoteRuneToASCII(r rune) string](https://pkg.go.dev/strconv@go1.19.4" \l "QuoteRuneToASCII)

## [func QuoteRuneToGraphic(r rune) string](https://pkg.go.dev/strconv@go1.19.4" \l "QuoteRuneToGraphic)

## [func QuoteToASCII(s string) string](https://pkg.go.dev/strconv@go1.19.4" \l "QuoteToASCII)

## [func QuoteToGraphic(s string) string](https://pkg.go.dev/strconv@go1.19.4" \l "QuoteToGraphic)

## [func QuotedPrefix(s string) (string, error)](https://pkg.go.dev/strconv@go1.19.4" \l "QuotedPrefix)

## [func Unquote(s string) (string, error)](https://pkg.go.dev/strconv@go1.19.4" \l "Unquote)

## [func UnquoteChar(s string, quote byte) (value rune, multibyte bool, tail string, err error)](https://pkg.go.dev/strconv@go1.19.4" \l "UnquoteChar)

# [type NumError](https://pkg.go.dev/strconv@go1.19.4" \l "NumError)

str := "Not a number"  
if \_, err := strconv.ParseFloat(str, 64); err != nil {  
 e := err.(\*strconv.NumError)  
 fmt.Println("Func:", e.Func) //Func: ParseFloat  
 fmt.Println("Num:", e.Num) //Num: Not a number  
 fmt.Println("Err:", e.Err) //Err: invalid syntax  
 fmt.Println(err)  
}

## [func (e \*NumError) Error() string](https://pkg.go.dev/strconv@go1.19.4" \l "NumError.Error)

## [func (e \*NumError) Unwrap() error](https://pkg.go.dev/strconv@go1.19.4" \l "NumError.Unwrap)