[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)/[Interchain Developer Academy](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)



Search

[Interchain Developer Academy](https://ida.interchain.io/)[Interchain Developer Academy](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

Search



Filters

Interchain Developer Academy

[](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Week 0 - Getting Started](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Getting Started](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Blockchain 101](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Blockchain History](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Public and Managed Blockchains](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Consensus in Distributed Networks](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Cryptography](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Self-Assessment Quiz](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Go Introduction - First Steps](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Go Basics](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Go Interfaces](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Control Structures in Go](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Arrays and Slices in Go](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Standard Packages in Go](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Concurrency in Go](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Good-To-Know Dev Terms](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Docker Introduction](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Week 1 - Introduction to the Interchain](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Introduction to the Interchain](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Blockchain Technology and the Interchain](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[The Interchain Ecosystem](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Getting ATOM and Staking It](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[A Blockchain App Architecture](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Accounts](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Transactions](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Messages](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Modules](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Protobuf](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Multistore and Keepers](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[BaseApp](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Queries](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Events](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Context](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Testing](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Relaying with IBC](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Interchain Security](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Bridges](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Migrations](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Week 1 Quiz](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Week 2 - First Steps](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[First Steps](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Setup Your Work Environment](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Run a Node, API, and CLI](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Ignite CLI](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Store Object](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Create Custom Messages](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Create and Save a Game Properly](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Add a Way to Make a Move](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Emit Game Information](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Record the Game Winner](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Week 2 Exercise](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Week 3 - Introduction to IBC and CosmJS](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Introduction to IBC and CosmJS](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[What is IBC?](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[IBC/TAO - Connections (OPTIONAL)](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[IBC/TAO - Clients (OPTIONAL)](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[IBC Token Transfer](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Interchain Accounts (OPTIONAL)](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[IBC Middleware (OPTIONAL)](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Create a Custom IBC Middleware (OPTIONAL)](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Integrate IBC Middleware Into a Chain (OPTIONAL)](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[IBC Tooling](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[What is CosmJS?](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Your First CosmJS Actions](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Compose Complex Transactions](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Learn to Integrate Keplr](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Create Custom CosmJS Interfaces](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Week 4 - Ignite CLI and IBC Advanced](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Ignite CLI and IBC Advanced](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Keep an Up-To-Date Game Deadline](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Keep Track Of How Many Moves Have Been Played](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Put Your Games in Order](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Auto-Expiring Games](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Let Players Set a Wager](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Handle wager payments](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Integration tests](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Incentivize Players](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Help Find a Correct Move](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Play With Cross-Chain Tokens](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Understand IBC Denoms](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Go Relayer](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Hermes Relayer](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Week 5 - CosmJS Advanced](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[CosmJS Advanced](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Create Custom Objects](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Create Custom Messages](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Get an External GUI](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Integrate CosmJS and Keplr](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Backend Script for Game Indexing](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Week 6 - IBC Deep Dive](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[IBC Deep Dive](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[IBC Application Developer Introduction](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Make a Module IBC-Enabled](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Adding Packet and Acknowledgment Data](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Create a Leaderboard Chain](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Week 7 - From Code to MVP to Production and Migrations](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[From Code to MVP to Production and Migrations](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Run in Production](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Prepare the Software to Run](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Prepare a Validator and Keys](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Prepare Where the Node Starts](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Prepare and Connect to Other Nodes](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Configure, Run, and Set Up a Service](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Prepare and Do Migrations](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Simulate Production in Docker](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Tally Player Info After Production](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Add a Leaderboard as a Module](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Migrate the Leaderboard Module After Production](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Simulate a Migration in Docker](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Final Exam](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[What's Next?](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

[Continue Your Interchain Journey](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html)

Docs Version Switcher

On this page

[fmt](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#fmt)

[strconv](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#strconv)

[Convert from string](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#convert-from-string)

[Convert to string](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#convert-to-string)

[Errors](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#errors)

[#Copy link](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#standard-packages-in-go) **Standard Packages in Go**

Like other languages, Go offers a lot of useful standard packages. You will look at:

* *fmt*
* *strconv*
* *errors*

[#Copy link](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#fmt) fmt

*fmt* (format) implements are formatted I/O - you covered a good number of its *verbs* in module 1. Now you will see a range of additional useful verbs:

* %% for the percent sign
* %#v for the Go-syntax representation
* %t for a boolean; it formats a value as true or false
* %b for an integer; it formats the integer to its binary representation
* %c for an integer; it formats the integer to its corresponding Unicode character

If you use an invalid verb, you will get a string beginning with "%!" and a description.

To write to stdout:

1. func Print(a ...interface{}) (n int, err error) formats with default formats.
2. func Println(a ...interface{}) (n int, err error) formats with default formats and appends a newline.
3. func Printf(format string, a ...interface{}) (n int, err error) formats with given format.

These three functions return the number of bytes written and any applicable error.

To read from stdin:

1. func Scan(a ...interface{}) (n int, err error) parses using default formats. Values are space-separated. Newlines count as spaces.
2. func Scanln(a ...interface{}) (n int, err error) parses using default formats and stops at a newline.
3. func Scanf(format string, a ...interface{}) (n int, err error) parses using the given format. The verb %c matches every rune (space, tab, newline, etc.).

These three functions return the number of items scanned and an error if the count of items parsed is smaller than the count of arguments.

Perhaps you noticed a kind of function declaration we did not speak about: the *variadic function* can be called with any number of arguments. You can iterate over the arguments as follows:



Copy

func variadicFunction(a ...interface{}) {

for \_,s:= range a {

// do something

}

}

There are formatting functions on *io.Writer* and *io.Reader*, so we also have:

1. func Fprint(w io.Writer, a...interface{}) (n int, err error): same as Print but writes to w.
2. func Fprintln(w io.Writer, a...interface{}) (n int, err error): same as Println but writes to w.
3. func Fprintf(w io.Writer, a...interface{}) (n int, err error): same as Printf but writes to w.
4. func Fscan(r io.Reader, a...interface{}) (n int, err error): same as Fscan but scans from r.
5. func Fscanln(r io.Reader, a...interface{}) (n int, err error): same as Fscanln but scans from r.
6. func Fscanf(r io.Reader, a...interface{}) (n int, err error): same as Fscanf but scans from r.

io.Writer is the interface that declares the Write method. io.Reader is also an interface and it declares the Read method.

The functions Sprint, Sprintln, and Sprintf are similar to Print, Println, and Printf, with the difference that they return a string instead of writing to stdout.

The functions Sscan, Scanln, and Sscanf are similar to Fscan, Fscanln, and Fscanf with the difference that they scan from a string given as an argument.

The function Errorf formats according to a format and returns it as an error.



**Further readings**

* [String formatting example (opens new window)↗](https://gobyexample.com/string-formatting)
* [Go walkthrough fmt (opens new window)↗](https://medium.com/go-walkthrough/go-walkthrough-fmt-55a14bbbfc53)
* [Regular expressions example (opens new window)↗](https://gobyexample.com/regular-expressions)
* [JSON example (opens new window)↗](https://gobyexample.com/json)
* [Io writer interface (opens new window)↗](https://medium.com/@as27/a-simple-beginners-tutorial-to-io-writer-in-golang-2a13bfefea02)

[#Copy link](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#strconv) strconv

The package *strconv* offers conversions to and from strings of basic data types.

[#Copy link](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#convert-from-string) Convert from string

Start with an example for using [strconv (opens new window)↗](https://golang.org/pkg/strconv):



Copy

package main

import (

"fmt"

"strconv"

)

func main() {

v32 := "-354634382"

if s, err := strconv.ParseInt(v32, 10, 32); err == nil {

fmt.Printf("%T, %v\n", s, s)

}

if s, err := strconv.ParseInt(v32, 16, 32); err == nil {

fmt.Printf("%T, %v\n", s, s)

}

v64 := "-3546343826724305832"

if s, err := strconv.ParseInt(v64, 10, 64); err == nil {

fmt.Printf("%T, %v\n", s, s)

}

if s, err := strconv.ParseInt(v64, 16, 64); err == nil {

fmt.Printf("%T, %v\n", s, s)

}

}



[Test it online (opens new window)↗](https://go.dev/play/p/GXstxF-6XVO).

* The function ParseInt(s string, base int, bitSize int) (i int64, err error) interprets s as an integer in base base, from 2 to 36; and with type bitSize, whereby 0 means int, 8 means int8, 16 means int16, 32 means int32, and 64 means int64.
* ParseUint is similar, with the difference that it returns an unsigned integer.
* The func Atoi(s string) (int, error) returns ParseInt(s, 10, 0) as type int.

[#Copy link](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#convert-to-string) Convert to string

1. func FormatBool(b bool) string returns "true" or "false" according to b.
2. func FormatFloat(f float64, fmt vyte, prec, bitSize int) string returns a string representation of f with format fmt and with precision prec.
3. func FormatInt(i int64, base int) string returns the string representation of i in base base.
4. func FormatUint(i uint, base int) string is the same as FormatInt, but it takes a unsigned integer.
5. func Itoa(i int) string is shorthand for FormatInt(int64(i), 10), so it gives a decimal representation of i as a string.



**Further reading:**

* [Go walkthrough - includes also string operations (opens new window)↗](https://gobyexample.com/string-formatting)

[#Copy link](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#errors) Errors

In Go, errors are values. The convention is that the last return value of a function is the error. This is the entire code of errors package:



Copy

// Copyright 2011 The Go Authors. All rights reserved.

// Use of this source code is governed by a BSD-style

// license that can be found in the LICENSE file.

// Package errors implements functions to manipulate errors.

package errors

// New returns an error that formats as the given text.

func New(text string) error {

return &errorString{text}

}

// errorString is a trivial implementation of error.

type errorString struct {

s string

}

func (e \*errorString) Error() string {

return e.s

}

You use func New(text string) error to create an error. An example is as follows:



Copy

package main

import (

"errors"

"fmt"

)

func div(a, b float64) (float64, error) {

if b == 0 {

return -1, errors.New("can't perform division by zero")

}

return a / b, nil

}

func main() {

for i := 2.; i >= -2; i = i - 0.5 {

if x, err := div(3, i); err != nil {

fmt.Println(err)

} else {

fmt.Printf("3/%v=%v\n", i, x)

}

}

}



[Test it online (opens new window)↗](https://go.dev/play/p/k9U_6uCkL1r).

In this case, nil means no error. For best practice, you should always check for errors. However, take the time to review the following list, because Go error handling differs from other languages.



**Further reading:**

* [A tour of Go (opens new window)↗](https://tour.golang.org/)

synopsis

To summarize, this section has explored:

* The ***fmt* (format)** standard package in increased detail, including a range of useful verbs and function declarations.
* The ***strconv*** package, which offers conversions to and from strings of basic data types.
* **Error** values, which by convention are the last return value of a function.

previous

[](https://ida.interchain.io/tutorials/4-golang-intro/5-arrays.html)

**[Arrays and Slices in Go](https://ida.interchain.io/tutorials/4-golang-intro/5-arrays.html)**

up next

**[Concurrency in Go](https://ida.interchain.io/tutorials/4-golang-intro/7-concurrency.html)**

[[](https://ida.interchain.io/tutorials/4-golang-intro/7-concurrency.html)](https://ida.interchain.io/tutorials/4-golang-intro/7-concurrency.html)

Rate this Page

icon smile

icon meh

icon frown

Would you like to add a message?

Submit

Thank you for your Feedback!

[](https://ida.interchain.io/ida-course/discord-info.html)

On this page

[fmt](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#fmt)

[strconv](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#strconv)

[Convert from string](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#convert-from-string)

[Convert to string](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#convert-to-string)

[Errors](https://ida.interchain.io/tutorials/4-golang-intro/6-packages.html#errors)

#### **Get Cosmos updates**

Unsubscribe at any time. [Privacy Policy↗](https://v1.cosmos.network/privacy)

     Next

Documentation

[Cosmos SDK](https://docs.cosmos.network/)[Cosmos Hub](https://hub.cosmos.network/)[CometBFT](https://docs.cometbft.com/)[IBC Protocol](https://ibc.cosmos.network/)

Community

[Interchain blog](https://blog.cosmos.network/)[Forum](https://forum.cosmos.network/)[Discord](https://discord.gg/cosmosnetwork)

Contributing

[Source code on GitHub](https://github.com/cosmos/sdk-tutorials)

[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)

**[](https://blog.cosmos.network/)[](https://twitter.com/cosmos)[](https://discord.gg/cosmosnetwork)[](https://www.linkedin.com/company/interchain-foundation/about/)[](https://reddit.com/r/cosmosnetwork)[](https://t.me/cosmosproject)[](https://www.youtube.com/c/CosmosProject)**



Dark mode

† This website is maintained by the Interchain Foundation (ICF). The contents and opinions of this website are those of the ICF. The ICF provides links to cryptocurrency exchanges as a service to the public. The ICF does not warrant that the information provided by these websites is correct, complete, and up-to-date. The ICF is not responsible for their content and expressly rejects any liability for damages of any kind resulting from the use, reference to, or reliance on any information contained within these websites.

Cosmos is a registered trademark of the [Interchain Foundation.](https://interchain.io/)[Privacy](https://v1.cosmos.network/privacy)