This page gives some hints on how to successfully ask for help in the various Go support forums.

Before you ask your question

Before asking for help, please check that you've addressed the following common issues:

Always check all errors

Always check all errors. It is common to see problems reported related to nil panics due to code like this

```
result, err := somefunction()
if err != nil {
      log.Println("oops an error happened", err)
      // return is missing here
}
// the code then continues to use result which is invalid.
```

or

```
result, _ := somefunction()
// code uses result which might be invalid
```

You should make sure it is clear that your code is correctly handling all error conditions before asking for help.

Further reading: - Error handling and Go

Check that your code is free from data races

Unexpected runtime panics are often caused by data races in your program. If your program contains a data race you need to address the race before asking for help.

If your program has good test coverage you can test for races by adding the `-race` flag to your `go test` invocation.

If your program does not have good test coverage or the crash only happens when running the program, you can build a race enabled version of your program by passing `-race` to your `go build` or `go install` invocation.

The behaviour of a Go program with a data race is undefined. There are no safe data races in Go programs.

Further reading: - Introducing the race detector

Asking questions

The best way to get help is to show

1. **What you did, ideally with a small complete, stand-alone, example.** If you ran a command, show the command that you ran. If your program failed, provide the source of the program that failed. If the program is too large, or you cannot share the source, instead provide a self contained, runnable example, that demonstrates the problem.
2. **What you expected to happen.** If you expected the command to complete successfully, say that. If you expected the program to produce a particular output, give an example of the output you expected.
3. **What happened instead.** If the command failed, include the full output of the failure, not just a single line that you though was the cause. If the program failed to produce the expected output, include what it did output.

Additional tips

- If you are posting the output of a command, paste the text, not a screenshot of the text. If it's actually an image, that's ok.
- If you are posting a large amount of output, you may consider using a pastebin or gist service.
- When posting code samples, use the Go playground (unless it is unavailable in your country).