Laís Lima

What I Learned coding a CLI

My first last open source contribution learnings

NEXT >

Agenda

What we'll discuss today

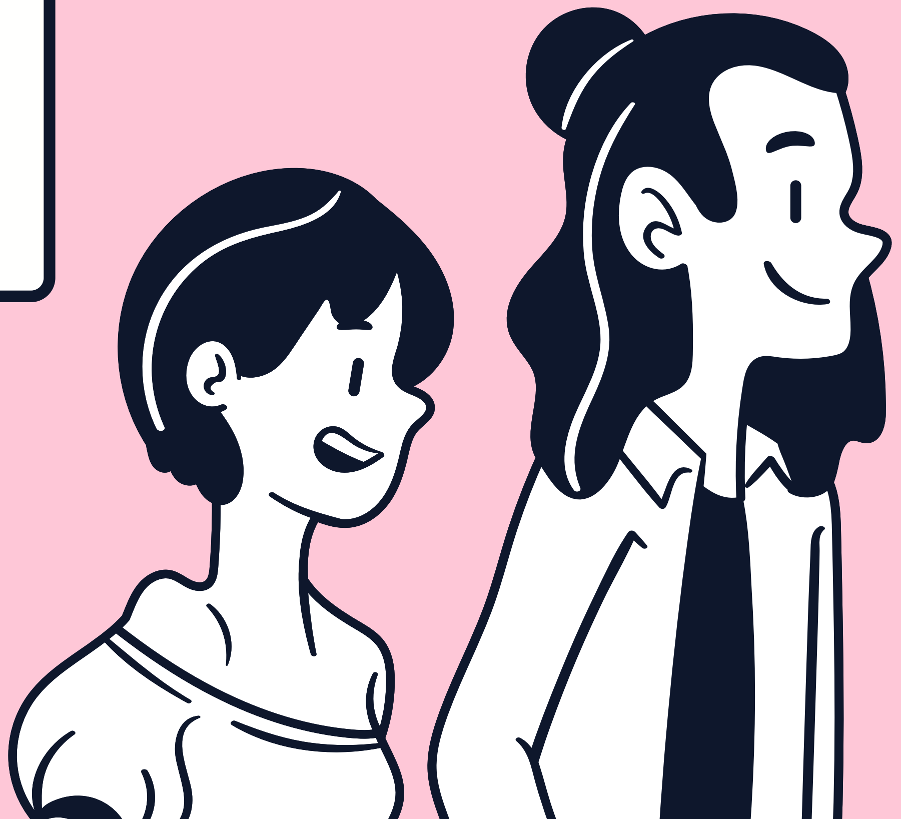
01 Golings Project

02 Building an iterative terminal

03 Challenges

04 Tears... Ops Conquers!

05 What I Learned



NEXT

About Me

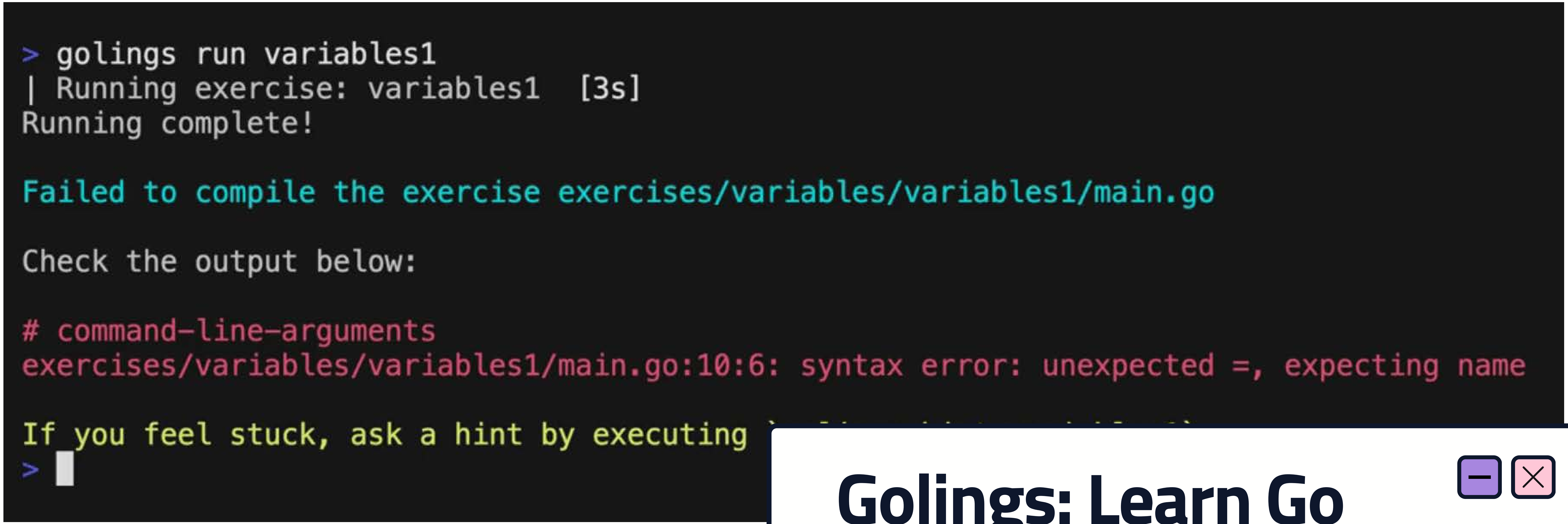
@laislima_dev

Laís Lima

Software Developer @ Globo
Guitarrist @ Tijolos Baianos

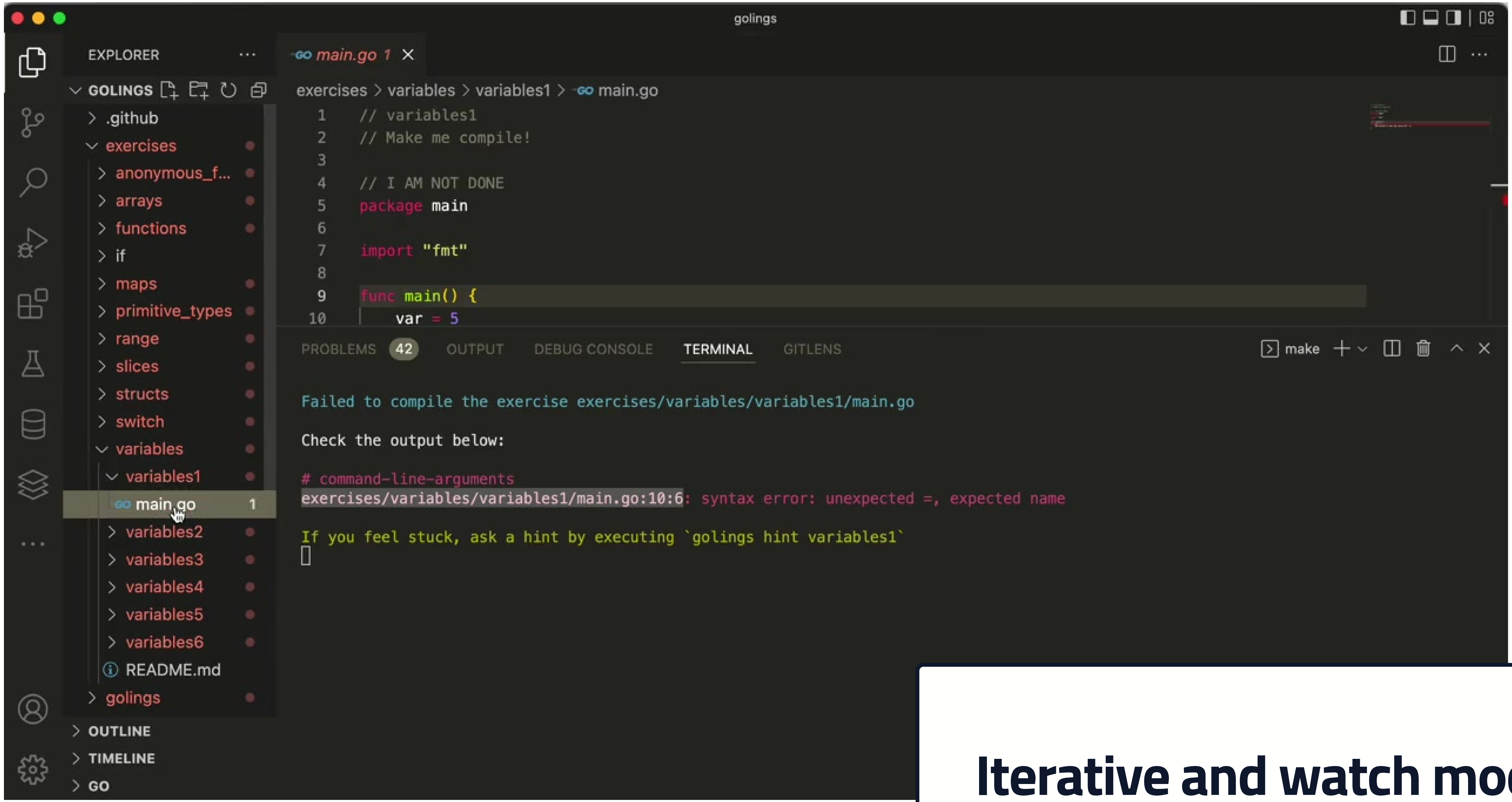
bio.link/laislima_dev

NEXT



```
> golings run variables1
| Running exercise: variables1  [3s]
Running complete!

Failed to compile the exercise exercises/variables/variables1/main.go

Check the output below:

# command-line-arguments
exercises/variables/variables1/main.go:10:6: syntax error: unexpected =, expecting name

If you feel stuck, ask a hint by executing
> 
```

Golings: Learn Go using a CLI



```go
// variables1
// Make me compile!

// I AM NOT DONE
package main

import "fmt"

func main() {
    var = 5
```

```
Failed to compile the exercise exercises/variables/variables1/main.go

Check the output below:

# command-line-arguments
exercises/variables/variables1/main.go:10:6: syntax error: unexpected =, expected name

If you feel stuck, ask a hint by executing `golings hint variables1`
```

Iterative and watch mode

What to do?

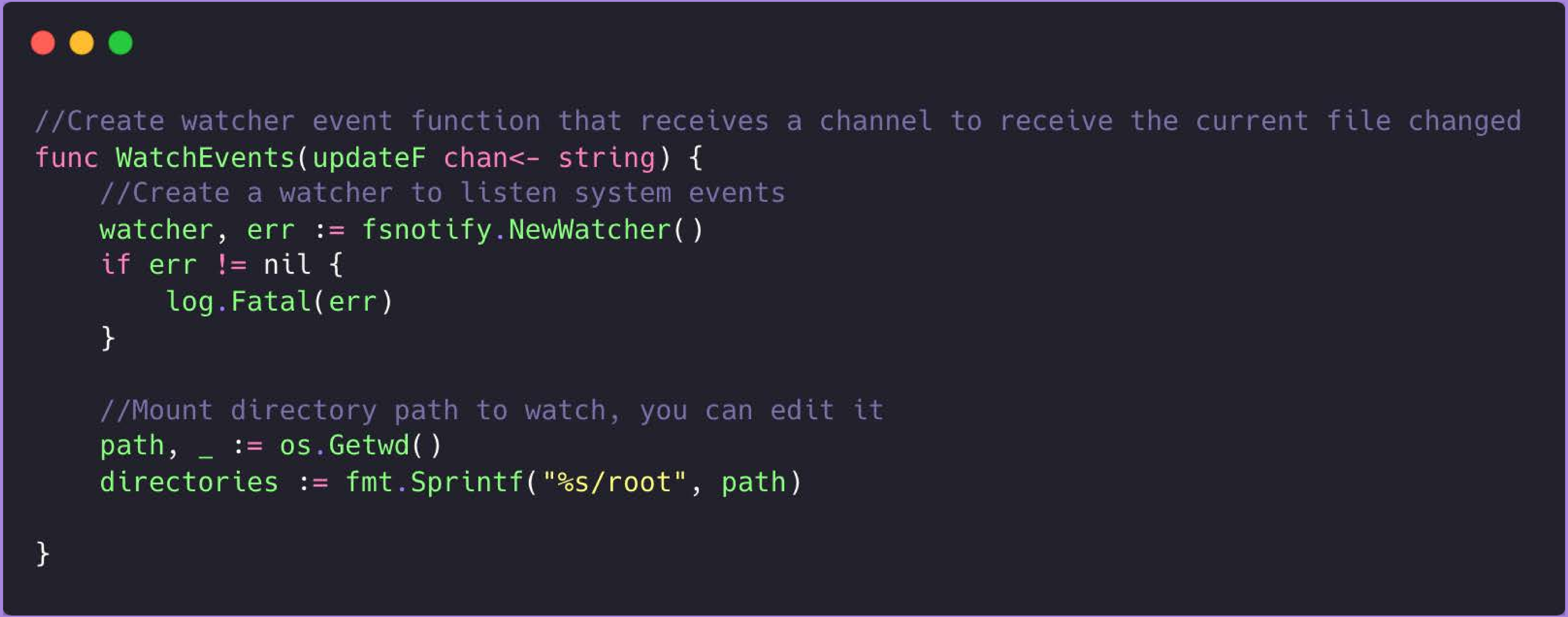
Listen to file changes

iterative mode with user inputs

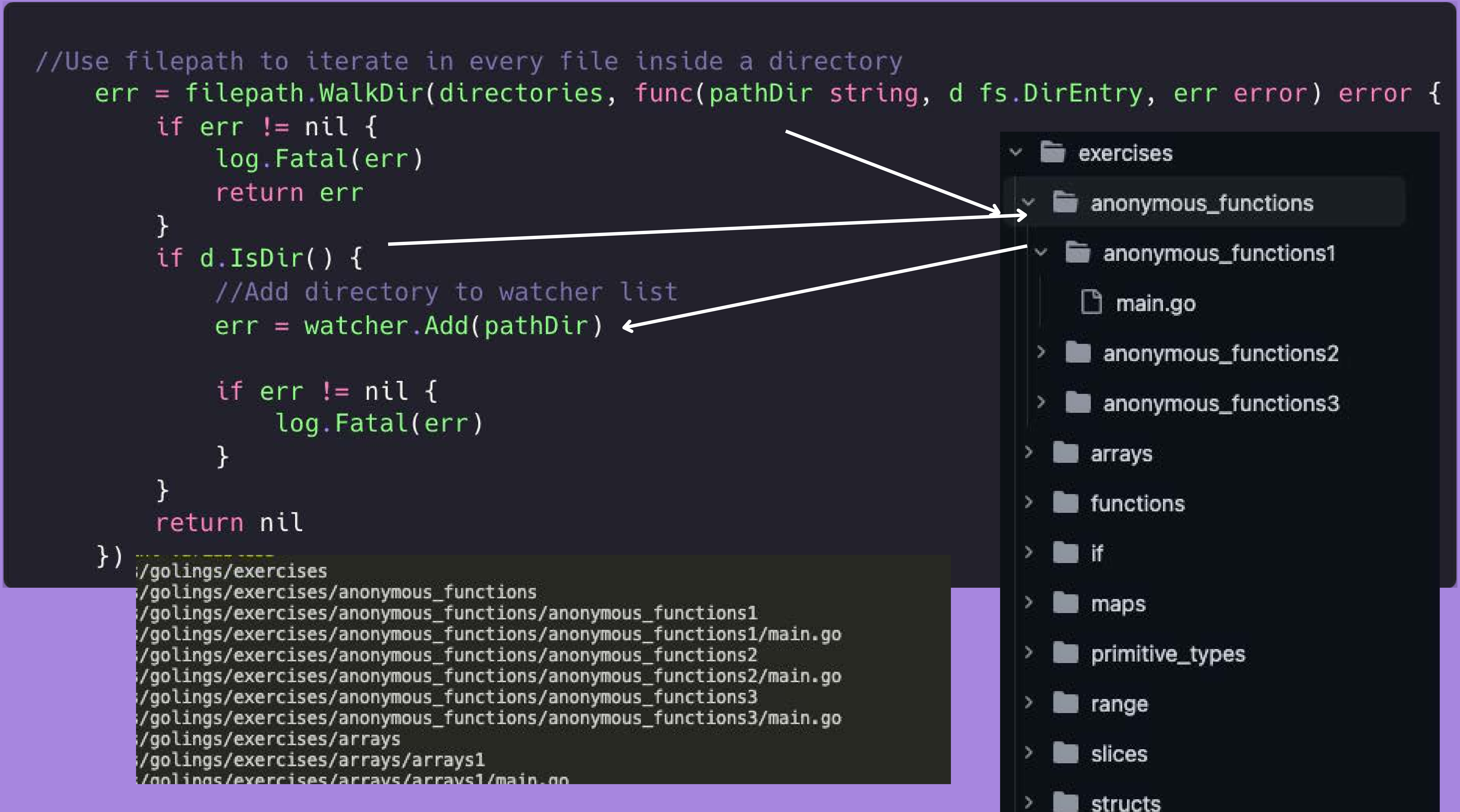
How to listen to file changes?

Infinite loops, goroutines and **fsnotify**.

NEXT



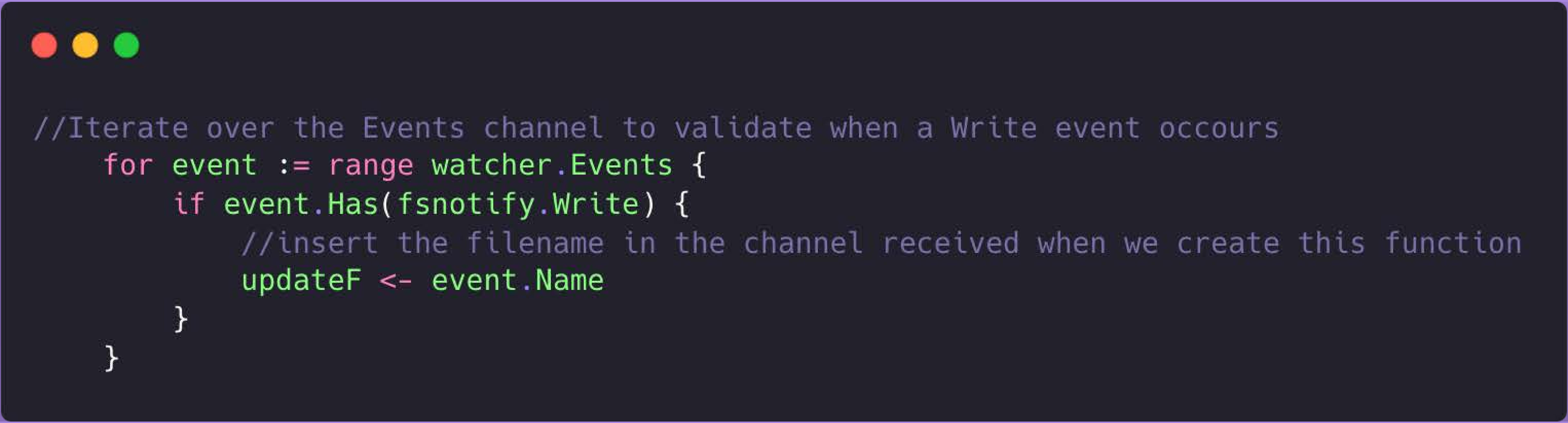
```go
//Create watcher event function that receives a channel to receive the current file changed
func WatchEvents(updateF chan<- string) {
    //Create a watcher to listen system events
    watcher, err := fsnotify.NewWatcher()
    if err != nil {
        log.Fatal(err)
    }

    //Mount directory path to watch, you can edit it
    path, _ := os.Getwd()
    directories := fmt.Sprintf("%s/root", path)

}
```



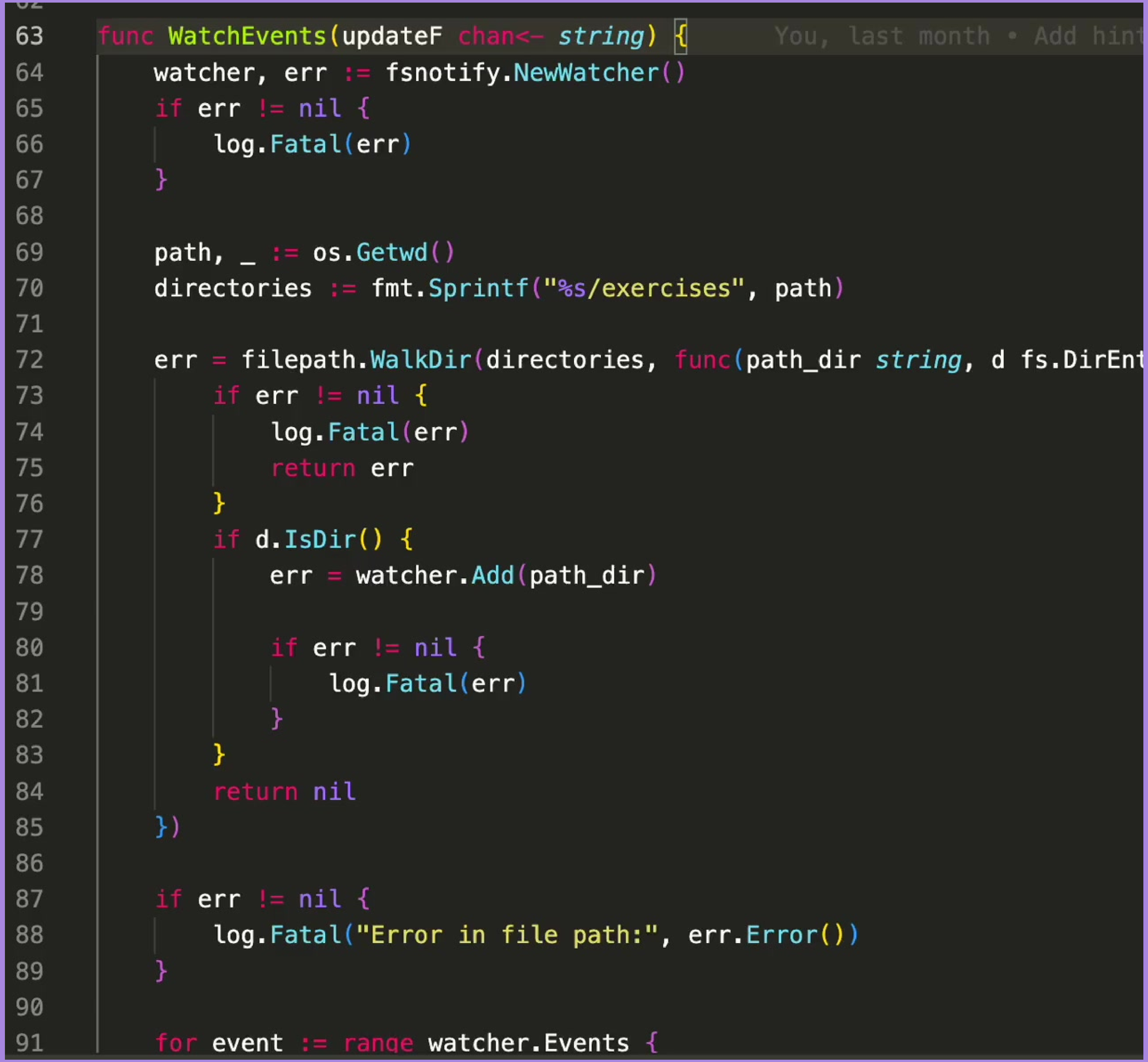
```go
//Use filepath to iterate in every file inside a directory
    err = filepath.WalkDir(directories, func(pathDir string, d fs.DirEntry, err error) error {
        if err != nil {
            log.Fatal(err)
            return err
        }
        if d.IsDir() {
            //Add directory to watcher list
            err = watcher.Add(pathDir)

            if err != nil {
                log.Fatal(err)
            }
        }
        return nil
    })
```

```
/golings/exercises
/golings/exercises/anonymous_functions
/golings/exercises/anonymous_functions/anonymous_functions1
/golings/exercises/anonymous_functions/anonymous_functions1/main.go
/golings/exercises/anonymous_functions/anonymous_functions2
/golings/exercises/anonymous_functions/anonymous_functions2/main.go
/golings/exercises/anonymous_functions/anonymous_functions3
/golings/exercises/anonymous_functions/anonymous_functions3/main.go
/golings/exercises/arrays
/golings/exercises/arrays/arrays1
/golings/exercises/arrays/arrays1/main.go
```

```
exercises
  anonymous_functions
    anonymous_functions1
      main.go
    anonymous_functions2
    anonymous_functions3
  arrays
  functions
  if
  maps
  primitive_types
  range
  slices
  structs
```



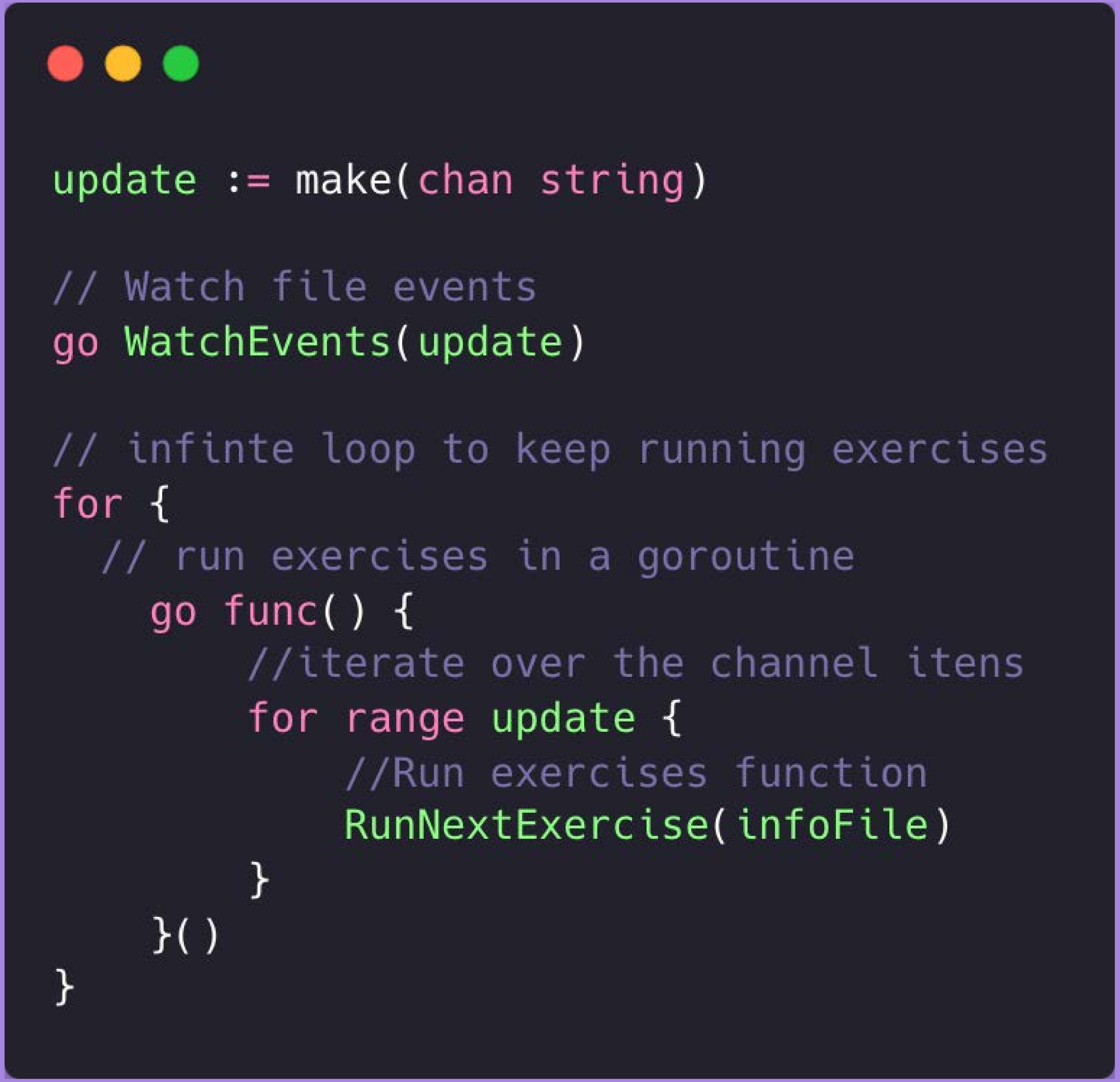
```
//Iterate over the Events channel to validate when a Write event occours
    for event := range watcher.Events {
        if event.Has(fsnotify.Write) {
            //insert the filename in the channel received when we create this function
            updateF <- event.Name
        }
    }
```



```
func WatchEvents(updateF chan<- string) {        You, last month • Add hint
    watcher, err := fsnotify.NewWatcher()
    if err != nil {
        log.Fatal(err)
    }

    path, _ := os.Getwd()
    directories := fmt.Sprintf("%s/exercises", path)

    err = filepath.WalkDir(directories, func(path_dir string, d fs.DirEnt
        if err != nil {
            log.Fatal(err)
            return err
        }
        if d.IsDir() {
            err = watcher.Add(path_dir)

            if err != nil {
                log.Fatal(err)
            }
        }
        return nil
    })

    if err != nil {
        log.Fatal("Error in file path:", err.Error())
    }

    for event := range watcher.Events {
```

Running every file changes

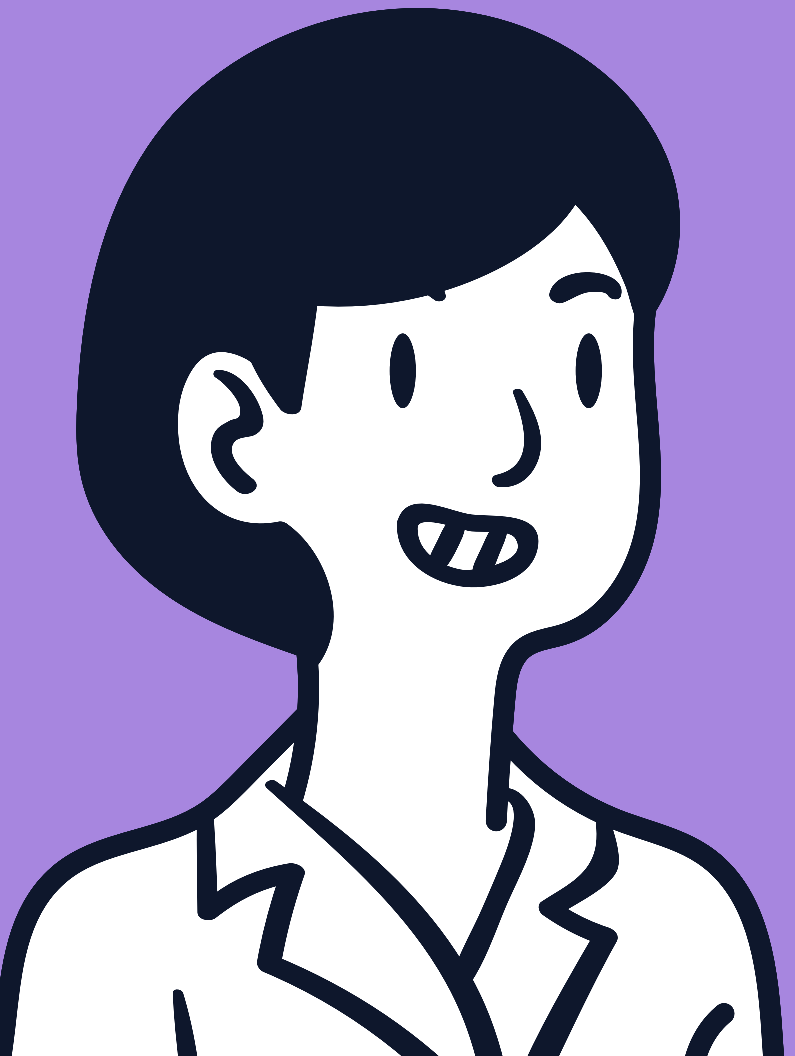
Run exercises every file changes



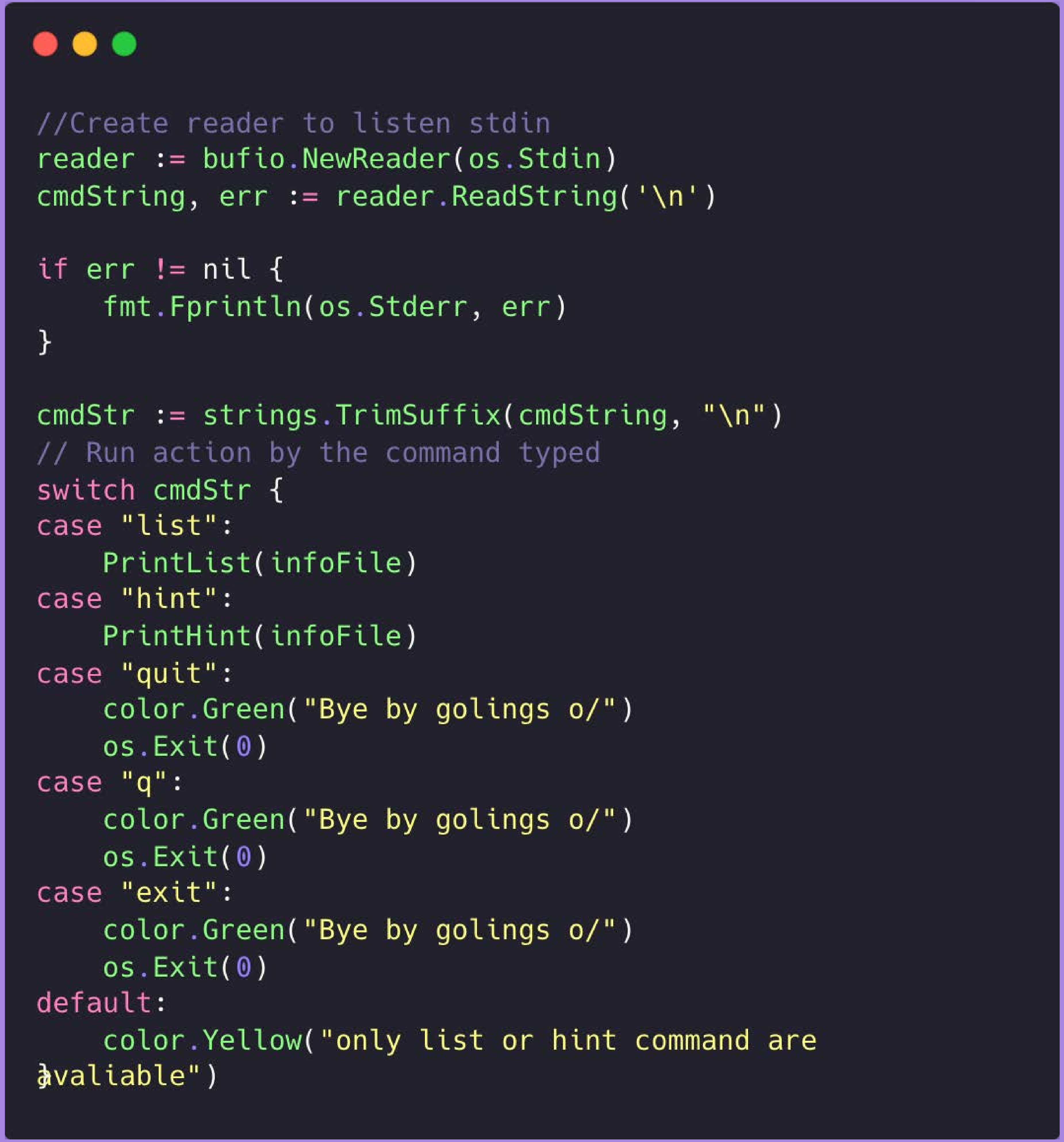
```go
update := make(chan string)

// Watch file events
go WatchEvents(update)

// infinte loop to keep running exercises
for {
  // run exercises in a goroutine
    go func() {
        //iterate over the channel itens
        for range update {
            //Run exercises function
            RunNextExercise(infoFile)
        }
    }()
}
```

Building the iterative mode

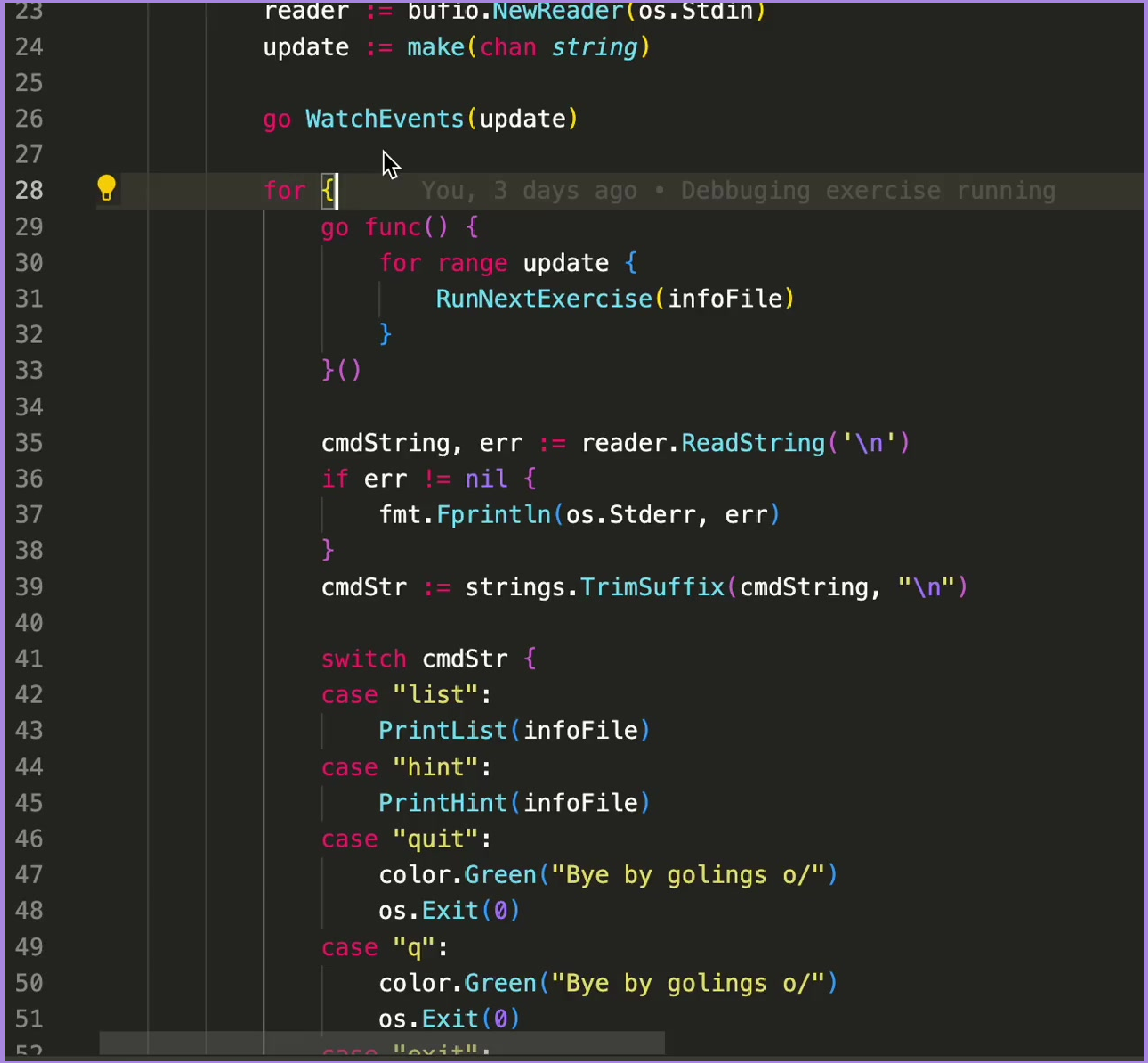
bufio packager and switch case inside an infinite loop



NEXT



```go
//Create reader to listen stdin
reader := bufio.NewReader(os.Stdin)
cmdString, err := reader.ReadString('\n')

if err != nil {
    fmt.Fprintln(os.Stderr, err)
}

cmdStr := strings.TrimSuffix(cmdString, "\n")
// Run action by the command typed
switch cmdStr {
case "list":
    PrintList(infoFile)
case "hint":
    PrintHint(infoFile)
case "quit":
    color.Green("Bye by golings o/")
    os.Exit(0)
case "q":
    color.Green("Bye by golings o/")
    os.Exit(0)
case "exit":
    color.Green("Bye by golings o/")
    os.Exit(0)
default:
    color.Yellow("only list or hint command are
}avaliable")
```



```go
    reader := bufio.NewReader(os.Stdin)
    update := make(chan string)

    go WatchEvents(update)

    for {      You, 3 days ago • Debbuging exercise running
        go func() {
            for range update {
                RunNextExercise(infoFile)
            }
        }()

        cmdString, err := reader.ReadString('\n')
        if err != nil {
            fmt.Fprintln(os.Stderr, err)
        }
        cmdStr := strings.TrimSuffix(cmdString, "\n")

        switch cmdStr {
        case "list":
            PrintList(infoFile)
        case "hint":
            PrintHint(infoFile)
        case "quit":
            color.Green("Bye by golings o/")
            os.Exit(0)
        case "q":
            color.Green("Bye by golings o/")
            os.Exit(0)
        case "exit":
```

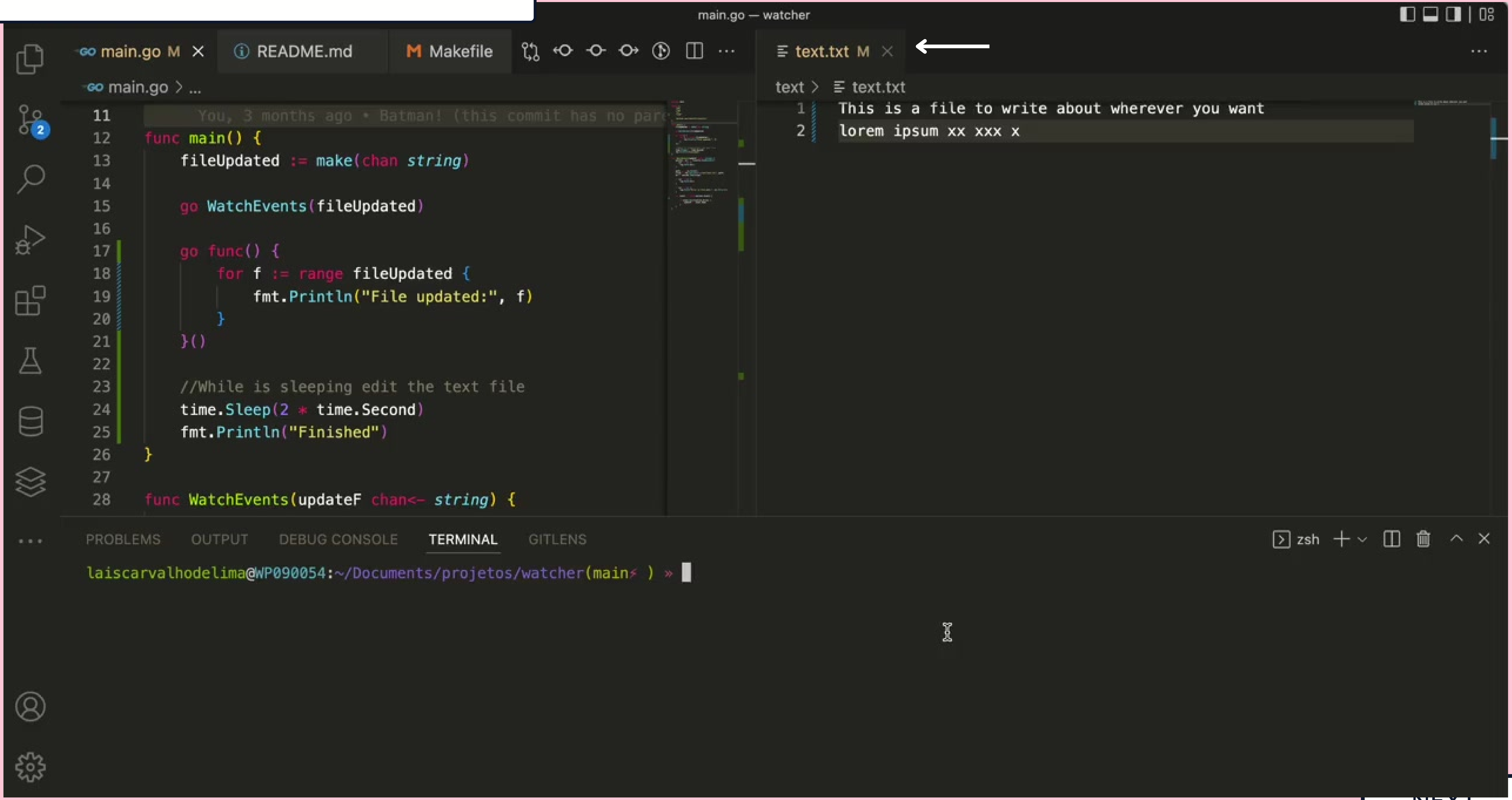
CHALLENGES

It's not too easy...

PROBLEM	GOLD
UNDERSTAND WHAT REALLY NEED TO DO	✓
TIME TO OPEN SOURCE CONTRIBUTION	✓
CONCURRENCY PROBLEMS	✓

NEXT

THE CONCURRENCY PROBLEM...



main.go — watcher

main.go M | README.md | Makefile | text.txt M

```go
	You, 3 months ago • Batman! (this commit has no pare
func main() {
	fileUpdated := make(chan string)

	go WatchEvents(fileUpdated)

	go func() {
		for f := range fileUpdated {
			fmt.Println("File updated:", f)
		}
	}()

	//While is sleeping edit the text file
	time.Sleep(2 * time.Second)
	fmt.Println("Finished")
}

func WatchEvents(updateF chan<- string) {
```

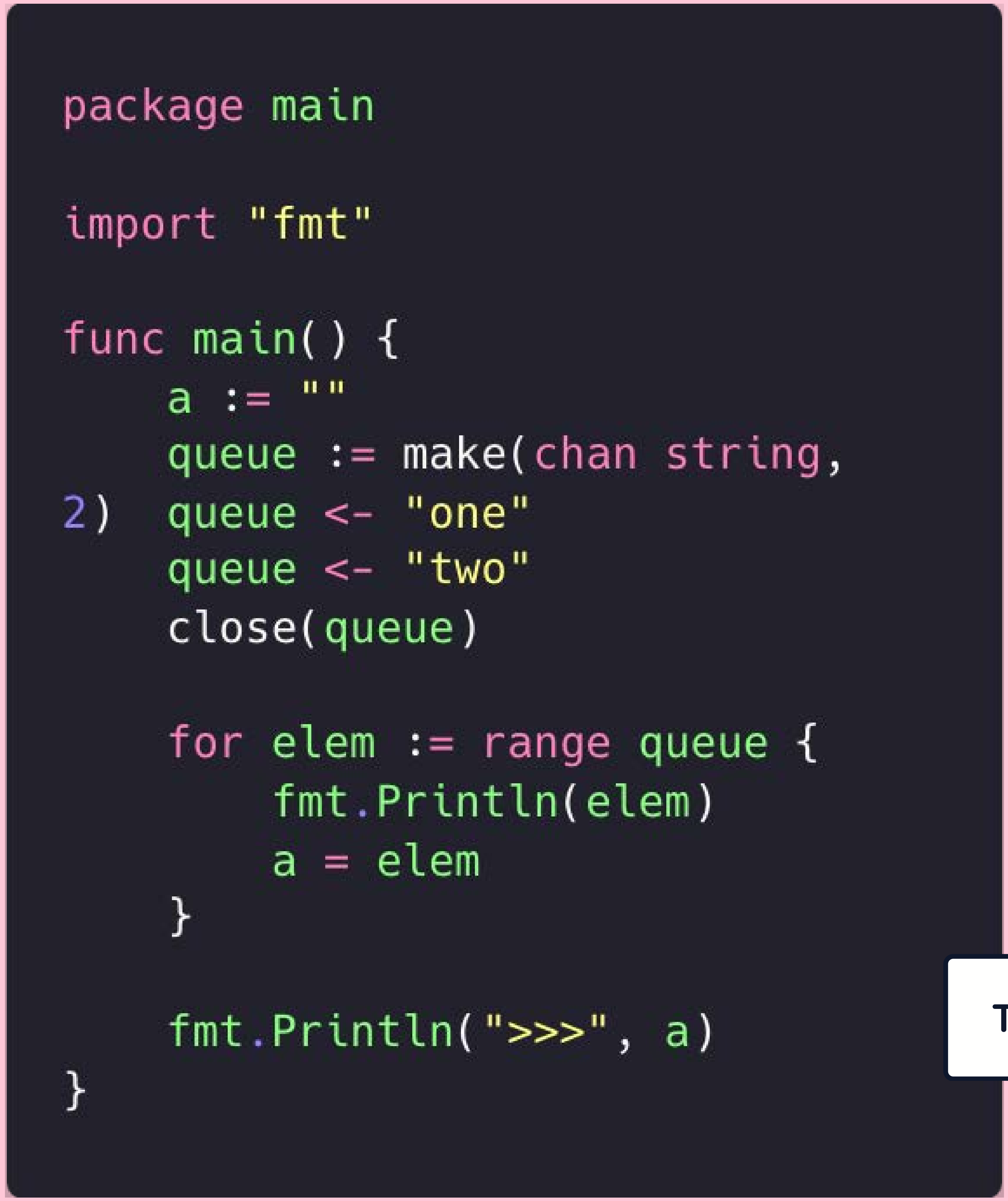
text > text.txt

```
This is a file to write about wherever you want
lorem ipsum xx xxx x
```

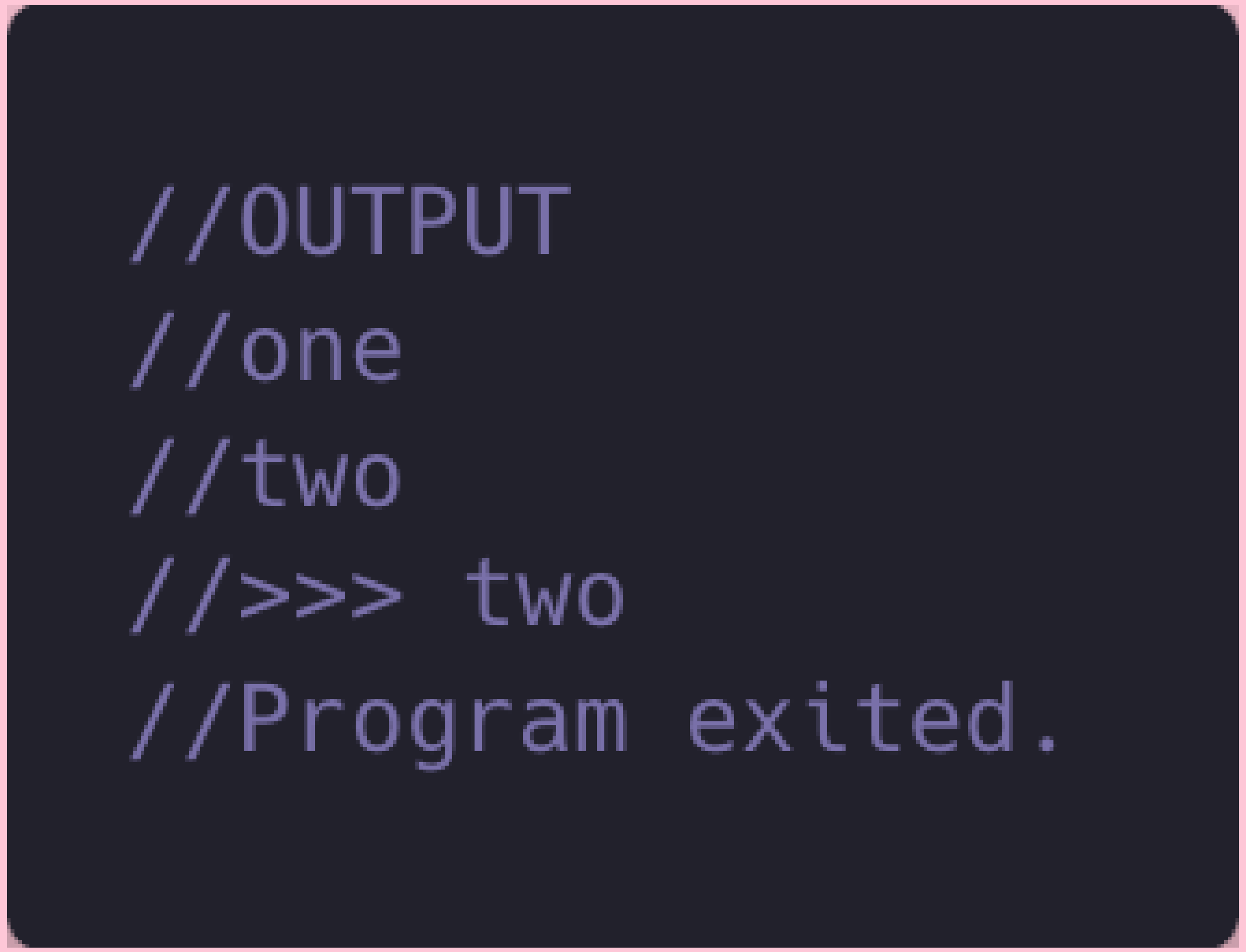
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS

```
laiscarvalhodelima@WP090054:~/Documents/projetos/watcher(main⚡ ) »
```

NEXT



```go
package main

import "fmt"

func main() {
    a := ""
    queue := make(chan string,
2)  queue <- "one"
    queue <- "two"
    close(queue)

    for elem := range queue {
        fmt.Println(elem)
        a = elem
    }

    fmt.Println(">>>", a)
}
```



```
//OUTPUT
//one
//two
//>>> two
//Program exited.
```

THE CONCURRENCY PROBLEM...

NEXT

WHAT I LEARNED

NEXT

WHAT I LEARNED

Feedbacks

Ask for feedbacks as soon as faster

NEXT

WHAT I LEARNED

Feedbacks

Ask for feedbacks as soon as faster

Go flow

Understand the Go flow

NEXT

WHAT I LEARNED

Feedbacks

Ask for feedbacks as soon as faster

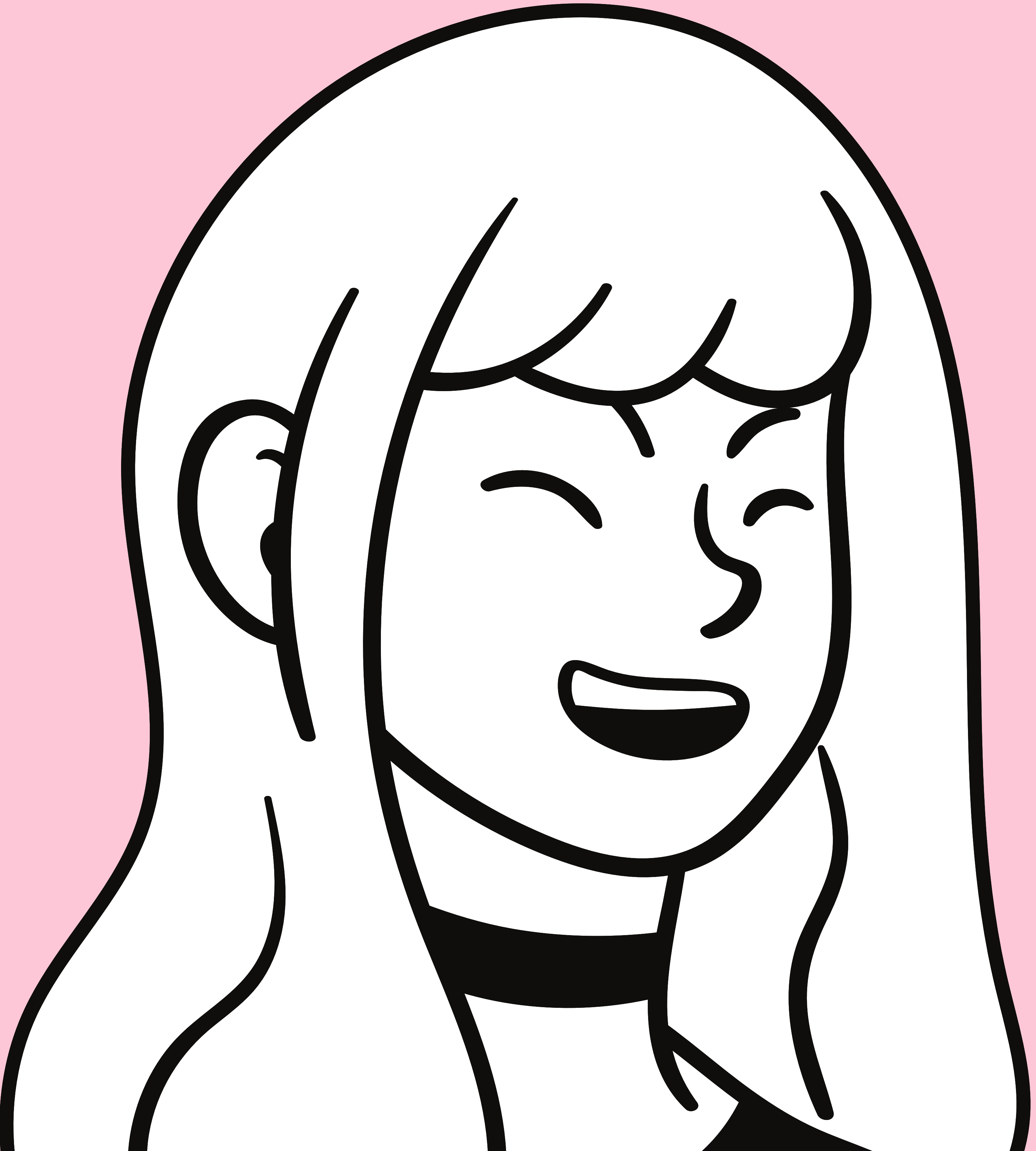
Go flow

Understand the Go flow

Open source

I will helps you!

NEXT



THANK YOU!

follow me @laislima_dev

bio.link/laislima_dev