



GO BRAIN TEASERS

EXERCISE YOUR MIND

```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     var  $\pi$  = 22 / 7.0
9     fmt.Println( $\pi$ )
10 }
```

WILL THIS CODE COMPILE? WHAT WILL IT PRINT?

25 MIND BENDING TEASERS & SOLUTIONS

MIKI TEBEKA

Copyright

Copyright © 2020, 353solutions LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

Are We There Yet?

time.go

```
1 package main
2
3 import (
4     "fmt"
5     "time"
6 )
7
8 func main() {
9     timeout := 3
10    fmt.Printf("before ")
11    time.Sleep(timeout * time.Millisecond)
12    fmt.Println("after")
13 }
```



Try to guess what the output is before moving to the next page.

This code will not compile.

When you write `timeout := 3` the Go compiler will do a [type inference](#). In this case it will infer that `timeout` is an `int`. Then in line 11 you multiply `timeout` (`int` type) with `time.Millisecond` (`time.Duration` type) - which is not allowed.

You have several options to fix this.

Change the type of `timeout` to `time.Duration` (line 9):

```
var timeout time.Duration = 3
```

Make `timeout` a const, and then its type will be resolved in the context of usage (line 9):

```
const timeout = 3
```

Use a [type conversion](#) to convert `timeout` to `time.Duration` (line 11):

```
time.Sleep(time.Duration(timeout) * time.Millisecond)
```

Further Reading

- [Type inference](#) in the Go tour
- [go/types: The Go Type Checker](#) in [golang/example](#)
- [time.Duration](#) type
- [Type conversions](#) in the Go specification