Go brain teasers

EXERCISE YOUR MIND

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

WILL THIS CODE COMPILE? WHAT WILL IT PRINT?

25 MIND BENDING TEASERS & SOLUTIONS

MIKI TEBEKA

Copyright

A Funky Number?

num.go

```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     fmt.Println(0x1p-2)
9 }
```

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

This code will print: `0.25`

Go has several number types, the two main ones are:

Integers

These are whole numbers. Go has `int8`, `int16`, `int32`, `int64` and `int`.[1]. There are also all the unsigned ones `uint8`...

Floats

These are real numbers. Go has `float32` and `float64`.

There are other types such as complex, and the various types defined in math/big.

When you write a number literal, such as `3.14`, the Go compiler needs to parse it to a specific type (`float64` in this case). The Go spec defines how you can write numbers. Let's have a look at some examples:

num_lit.go

```go
package main

import (
    "fmt"
)

func main() {
    // Integer
    printNum(10)    // 10 of type int
    printNum(010)   // 8 of type int
    printNum(0x10)  // 16 of type int
    printNum(0b10)  // 2 of type int
    printNum(1_000) // 1000 of type int ①

    // Float
    printNum(3.14)   // 3.14 of type float64
    printNum(.2)     // 0.2 of type float64
    printNum(1e3)    // 1000 of type float64 ②
    printNum(0x1p-2) // 0.25 of type float64 ③

    // Complex
    printNum(1i)     // (0+1i) of type complex128
    printNum(3 + 7i) // (3+7i) of type complex128
    printNum(1 + 0i) // (1+0i) of type complex128
}

func printNum(n interface{}) {
    fmt.Printf("%v of type %T\n", n, n)
}
```

① _ serves as the thousands separator. It makes big numbers much more readable for us humans.

② This is known as scientific notation

③ The current brain teaser

`0x1p-2` is called "a hexadecimal floating-point literal" in the Go specification and is following the IEEE 754 2008 specification. To calculate the value:

- Compute the value before the `p` as a hexadecimal number. In this example it's: `0x1` [2] = 1
- Compute the value after the `p` as "2 to the power of that value". In this example it's: 2^{-2} = 0.25 [3]
- Finally multiply the two numbers. In this example: `1 * 0.25` = 0.25

Further Reading

- IEEE 754 specification
- Integer literals
- Floating point literals
- Imaginary literals

[1] `int` is an alias to your system integer size, 64 on my machine.

[2] One in hexadecimal.

[3] Remember your high school math: $2^{-2} = 1/2^2$.