An Ultimate GopherLabs Hands-on Labs

Who Am I?

Sangam Biradar

Sangam Biradar

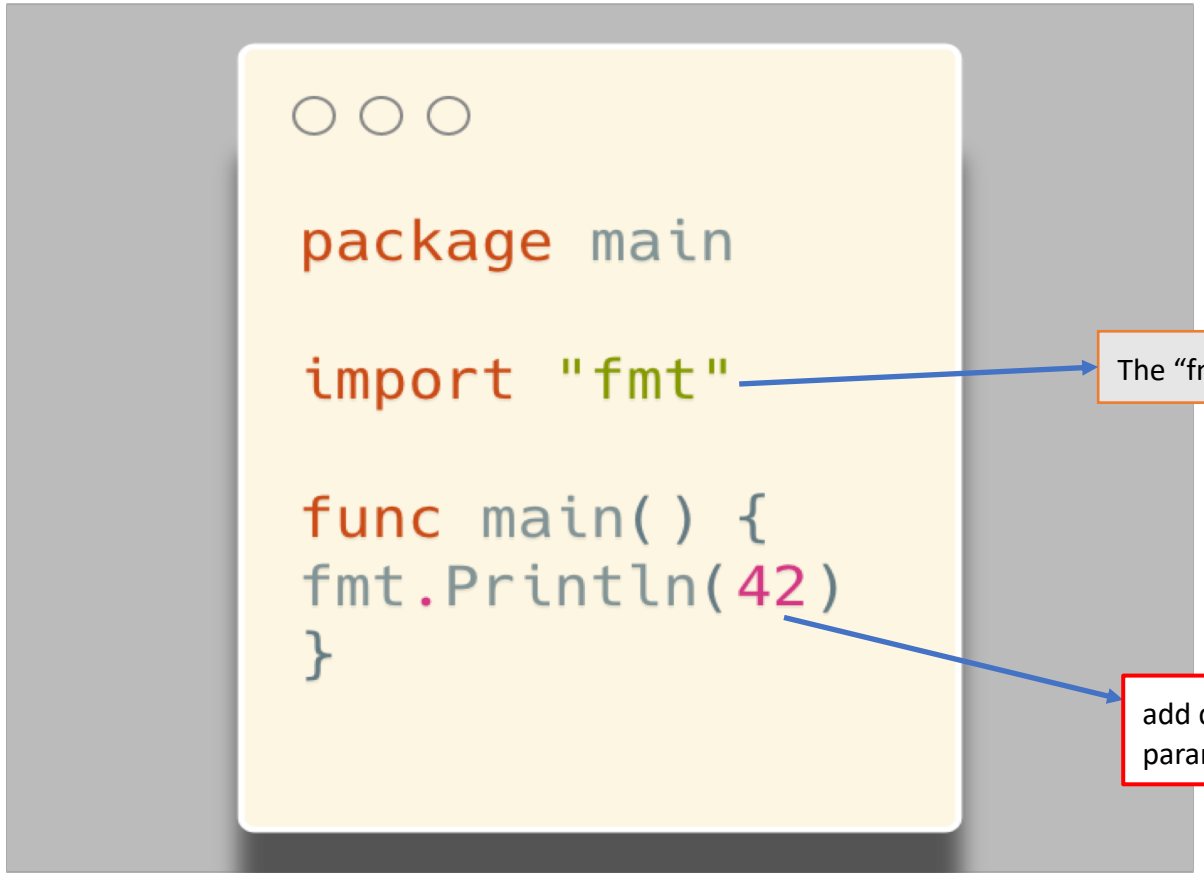
@BiradarSangam

EngineITops.com

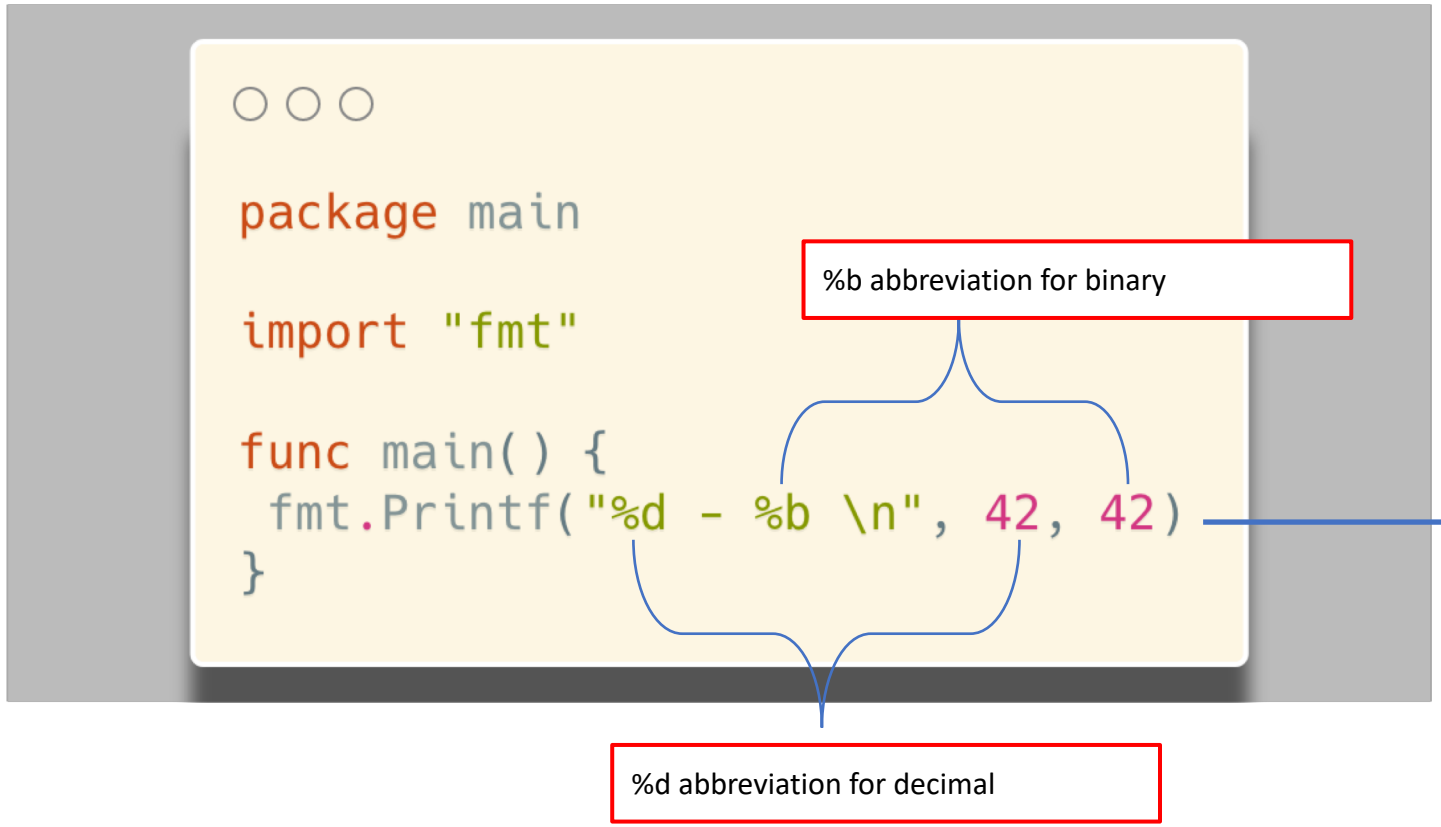
- Docker Community Leader , Bangalore
- Author :- lightweight Kubernetes with k3s with packt Publication
- Gopherlabs – 200+ tutorials
- Okteto – Kubernetes For Developer , Bangalore

Meetup Organizer

• Number System : Decimal



• Number System : binary



https://play.golang.org/p/8oA-JOz0flp

Output:
42 - 101010

• cheat sheet for number system

```
%b    base 2
%c    the character represented by the corresponding Unicode code point
%d    base 10
%o    base 8
%q    a single-quoted character literal safely escaped with Go syntax.
%x    base 16, with lower-case letters for a-f
%X    base 16, with upper-case letters for A-F
%U    Unicode format: U+1234; same as "U+%04
```

• Number System : Hexadecimal

%x base 16, with lower-case letters for a-f
%X base 16, with upper-case letters for a-f

```go
package main

import "fmt"

func main() {
    //  fmt.Printf("%d - %b - %x \n", 42, 42, 42)
    //  fmt.Printf("%d - %b - %#x \n", 42, 42, 42)
    //  fmt.Printf("%d - %b - %#X \n", 42, 42, 42)
    fmt.Printf("%d \t %b \t %#X \n", 42, 42, 42)
}
```

https://play.golang.org/p/Gp3DWfsu7hX

Output:-
42 - 101010 - 2a
42 - 101010 - 0x2a
42 - 101010 - 0X2A
42 101010 0X2A

• Number System – loop

```go
package main

import "fmt"

func main() {
    for i := 1; i < 16; i++ {
        fmt.Printf("%d \t %b \t %x \n", i, i, i)
    }
}
```

Output 👇

```
1    1    1
2    10       2
3    11       3
4    100      4
5    101      5
6    110      6
7    111      7
8    1000     8
9    1001     9
10   1010     a
11   1011     b
12   1100     c
13   1101     d
14   1110     e
15   1111     f
```

👉 https://play.golang.org/p/TxcV48laVMk

• UTF 8 = Unicode Transformation Format – 8-bit

UTF8 is a character encoding where is assigns 1,112,064 characters a binary number that is from 1 byte (8 bits) to 4 bytes (32 bits) long.

Why is it necessary? Well, our computers use binary to store data. So inside a computer information is a sequence of 0’s and 1’s. When you are writing a text file on your computer the computer needs to store that in binary code (in 0’s and 1’s). But what would the character 'a' represent in binary? Short answer is whatever you want it to be. That is why we have UTF8 (and ASCII before it), it provides a standard that says the letter 'a' will take the value of 01100010 in binary. It allows us to say this file is stored with the UTF8 encoding, so the binary code must be interpreted with that in mind.

• UTF-8

%q a single-quoted character literal safely escaped with Go syntax.

```go
package main

import "fmt"

func main() {
    for i := 60; i < 122; i++ {
        fmt.Printf("%d \t %b \t %x \t %q \n", i, i, i, i)
    }
}
```

https://play.golang.org/p/pDZK6lHdtfL

Output 👇

```
60   111100      3c      '<'
61   111101      3d      '='
.......
.......
.....
.............
100      1100100     64      'd'
101      1100101     65      'e'
102      1100110     66      'f'
103      1100111     67      'g'
104      1101000     68      'h'
105      1101001     69      'i'
106      1101010     6a      'j'
107      1101011     6b      'k'
108      1101100     6c      'l'
109      1101101     6d      'm'
110      1101110     6e      'n'
111      1101111     6f      'o'
112      1110000     70      'p'
113      1110001     71      'q'
114      1110010     72      'r'
115      1110011     73      's'
116      1110100     74      't'
117      1110101     75      'u'
118      1110110     76      'v'
119      1110111     77      'w'
120      1111000     78      'x'
121      1111001     79      'y'
```

References

- https://gopherlabs.collabnix.com
- https://godoc.org/
- https://golang.org/doc/

Thanks!

Any questions?

 Sangam Biradar

 @BiradarSangam

 @sangambiradar