



Mastertech, AUG 16 2019

# Basics Syntax #1




**GoFz**

---

Mastertech

---

Engineer, R&D

A photograph of a modern workspace featuring a silver laptop and a white mouse on a dark wooden desk. A white chair is partially visible in the foreground. The text 'Go is a modern, general purpose language.' is overlaid in a large, white, sans-serif font.

Go is a modern,  
general purpose  
language.

Compiles to native machine code (32-bit and 64-bit x86, ARM).

# Lightweight syntax.

“

Go's purpose is to  
make its designers'  
programming lives better.



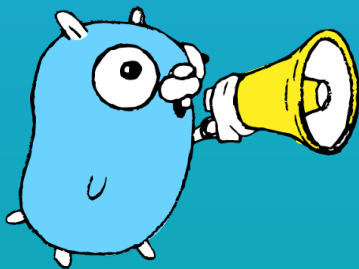
ROB PIKE

”









Hi, Let's talk about

|                         |    |
|-------------------------|----|
| Variables               | 01 |
| Looping ( For )         | 02 |
| Condition ( If / Else ) | 03 |
| Array                   | 04 |
| Slice                   | 05 |
| Map                     | 06 |
| Function                | 07 |
| Defer                   | 08 |



## THE GO PLAYGROUND

---

<https://play.golang.org>



```
package main

import (
    "fmt"
)

func main() {
    fmt.Println("Hello, ชาวโลก")
}
```

```
package main
```

```
import "fmt"
```

```
import "math"
```

```
func main() {
```

```
    fmt.Println("result =", math.Sqrt(7))
```

```
}
```

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    fmt.Println("random :", rand.Intn(10))
}
```

01

---

# Variables

bool

string

int   int8   int16   int32   int64  
uint   uint8   uint16   uint32   uint64

float32   float64

## Declare variables

---

```
package main

import (
    "fmt"
)

func main() {
    var a, b, c bool
    fmt.Println(a, b, c)
}
```

## Declare variables (summary)

---

```
package main

import (
    "fmt"
)

func main() {
    var a, b, c bool
    var d, e, f bool = true, false, true
    var g, h, i = false, 10, "hello"
    j, k, l := 14.5, false, "apple"
    fmt.Println(a, b, c, d, e, f, g, h, i, j, k, l)
}
```



```
package main

import (
    "fmt"
)

func main() {
    var a int8 = 2
    var b int64 = 3
    fmt.Println("a + b =", a + b)
}
```

02

---

# Looping (for)

## For Loop

---



```
package main

import (
    "fmt"
)

func main() {
    for i := 0; i < 10; i++ {
        fmt.Println(i)
    }
}
```

# While - Do Loop

---

```
package main

import (
    "fmt"
)

func main() {
    i := 0
    for i < 10 {
        fmt.Println(i)
        i += 1
    }
}
```

# Do - While Loop



```
package main

import (
    "fmt"
)

func main() {
    i := 0
    for {
        fmt.Println(i)
        if i >= 10 {
            break
        }
        i += 1
    }
}
```

## Foreach ( range ) with Array

---

```
package main

import (
    "fmt"
)

func main() {
    arr := [5]int{1, 2, 3, 4, 5}
    for index, element := range arr {
        fmt.Println(index, element)
    }
}
```

## Foreach ( range ) with Map

---

```
package main

import (
    "fmt"
)

func main() {
    m := map[string]int{"foo": 1, "bar": 2}
    for key, value := range m {
        fmt.Println(key, value)
    }
}
```



## Foreach ( range ) with String

---



```
package main

import (
    "fmt"
)

func main() {
    s := "Hello"
    for index, element := range s {
        fmt.Println(index, element)
    }
}
```

03

---

# Condition ( if / else )

```
package main

import (
    "fmt"
)

func main() {
    if 8%4 == 0 {
        fmt.Println("8 is divisible by 4")
    }
}
```

## IF - ELSE

---



```
package main

import (
    "fmt"
)

func main() {
    if 7%2 == 0 {
        fmt.Println("7 is even")
    } else {
        fmt.Println("7 is odd")
    }
}
```

---

# IF - ELSE - IF



```
package main

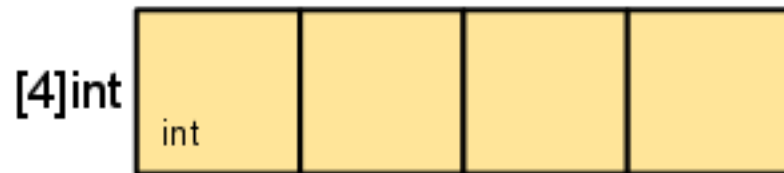
import (
    "fmt"
)

func main() {
    if num := 9; num < 0 {
        fmt.Println(num, "is negative")
    } else if num < 10 {
        fmt.Println(num, "has 1 digit")
    } else {
        fmt.Println(num, "has multiple digits")
    }
}
```

04

---

# Array





## Create an Array

---



```
package main

import (
    "fmt"
)

func main() {
    var a [5]int
    fmt.Println("emp:", a)
}
```

---

## Set & Get a value with index

---



```
package main

import (
    "fmt"
)

func main() {
    var a [5]int
    a[4] = 100
    fmt.Println("arr:", a)
    fmt.Println("get:", a[4])
}
```

## Declare & Initialize an Array

---

```
package main

import (
    "fmt"
)

func main() {
    a := [5]int{1, 2, 3, 4, 5}
    fmt.Println("arr:", a)
}
```

## Let's compiler count the array elements for you

---



```
package main

import (
    "fmt"
)

func main() {
    a := [...]int{1, 2, 3, 4, 5}
    fmt.Println("arr:", a)
}
```

## Length of an Array

---



```
package main

import (
    "fmt"
)

func main() {
    a := [...]int{2, 4, 6, 8, 10, 12}
    fmt.Println("len:", len(a))
}
```

# Multi-Dimensional Array (1)



```
package main

import (
    "fmt"
)

func main() {
    var twoD [2][3]int
    for i := 0; i < 2; i++ {
        for j := 0; j < 3; j++ {
            twoD[i][j] = i + j
        }
    }
    fmt.Println("2d: ", twoD)
}
```

## Multi-Dimensional Array (2)

---

```
package main

import (
    "fmt"
)

func main() {
    var twoD = [2][3]int{
        {0, 1, 2},
        {3, 4, 5},
    }
    fmt.Println("2d: ", twoD)
}
```



## Foreach ( range ) with Array

---

```
package main

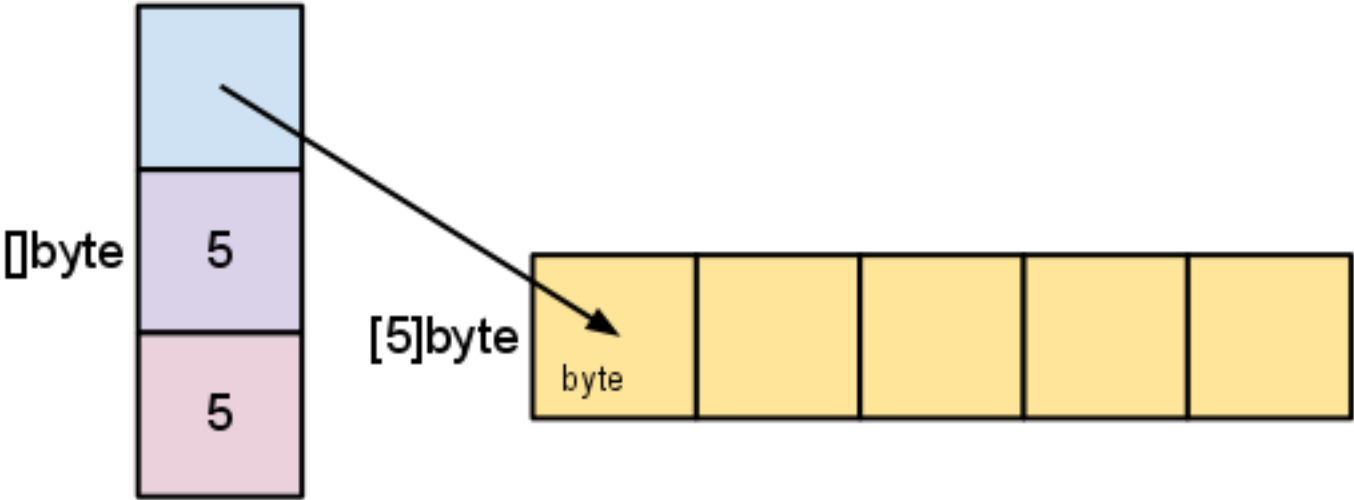
import (
    "fmt"
)

func main() {
    arr := [5]int{1, 2, 3, 4, 5}
    for index, element := range arr {
        fmt.Println(index, element)
    }
}
```

05

---

# Slice



```
package main

import (
    "fmt"
)

func main() {
    s := make([]int, 3)
    fmt.Println("emp:", s)
}
```

## Set & Get a value

---



```
package main

import (
    "fmt"
)

func main() {
    s := make([]int, 3)
    s[1] = 5
    fmt.Println("slice:", s)
    fmt.Println("get:", s[1])
}
```

## Append a new value

---



```
package main

import (
    "fmt"
)

func main() {
    s := make([]int, 3)
    s = append(s, 5)
    fmt.Println("slice:", s)
}
```

## Create a empty Slice

---



```
package main

import (
    "fmt"
)

func main() {
    s := []int{}
    fmt.Println("slice:", s)
}
```

---

## Declare & Initialize a Slice

---

```
package main

import (
    "fmt"
)

func main() {
    s := []int{1, 2, 3, 4, 5}
    fmt.Println("slice:", s)
}
```



## Length of a Slice

---

```
package main

import (
    "fmt"
)

func main() {
    s := []int{1, 2, 3, 4, 5}
    fmt.Println("len:", len(s))
}
```

## Copy a Slice

---



```
package main

import (
    "fmt"
)

func main() {
    s := []int{1, 2, 3, 4, 5}
    c := make([]int, len(s))
    copy(c, s)
    fmt.Println("copy:", c)
}
```

# Slice operator



```
package main

import (
    "fmt"
)

func main() {
    s := []int{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
    fmt.Println("s: ", s)
    fmt.Println("s1:", s[:])
    fmt.Println("s2:", s[2:])
    fmt.Println("s3:", s[:5])
    fmt.Println("s4:", s[2:5])
}
```

```
package main

import (
    "fmt"
)

func main() {
    s := []int{}
    s = append(s, 1)
    s = append(s, 2)
    s = append(s, 3)
    fmt.Println("slice:", s)
}
```

## Foreach ( range ) with Slice

---

```
package main

import (
    "fmt"
)

func main() {
    s := []int{2, 4, 6, 8, 10}
    for index, element := range s {
        fmt.Println(index, element)
    }
}
```

06

---

# Map

## Create a Map (1)

---

```
package main

import (
    "fmt"
)

func main() {
    m := make(map[string]int)
    fmt.Println("map:", m)
}
```

## Create a Map (2)

---

```
package main

import (
    "fmt"
)

func main() {
    m := map[string]int{}
    fmt.Println("map:", m)
}
```

---



## Set & Get Key / Value

---



```
package main

import (
    "fmt"
)

func main() {
    m := make(map[string]int)
    m["one"] = 1
    m["two"] = 2
    fmt.Println("value:", m["two"])
}
```

## Optional second return value

---

```
package main

import (
    "fmt"
)

func main() {
    m := make(map[string]int)
    m["one"] = 1
    m["two"] = 2
    v1, v2 := m["one"]
    fmt.Println("v1:", v1, "    v2:", v2)
}
```

## Length of a Map

---



```
package main

import (
    "fmt"
)

func main() {
    m := make(map[string]int)
    m["one"] = 1
    m["two"] = 2
    fmt.Println("len:", len(m))
}
```

## Remove key / value from a Map

---



```
package main

import (
    "fmt"
)

func main() {
    m := make(map[string]int)
    m["one"] = 1
    m["two"] = 2
    fmt.Println("map:", m)
    delete(m, "two")
    fmt.Println("map:", m)
}
```

## Foreach ( range ) with Map

---



```
package main

import (
    "fmt"
)

func main() {
    m := map[string]int{"foo": 1, "bar": 2}
    for key, value := range m {
        fmt.Println(key, value)
    }
}
```

07

---

# Function

```
package main

import "fmt"

func hello() {
    fmt.Println("สวัสดีชาวโลก")
}

func main() {
    hello()
}
```

## Function with parameter(s)

---

```
package main

import "fmt"

func plus(a int, b int) {
    fmt.Println("plus :", a+b)
}

func main() {
    plus(1, 2)
}
```



## Variadic Functions ( Function with infinite parameters )

---



```
package main

import "fmt"

func show(a ...int) {
    for i, v := range a {
        fmt.Println("index:", i, "value:", v)
    }
}

func main() {
    show(1, 2, 3)
}
```

## Return Values

---



```
package main

import "fmt"

func plus(a int, b int) int {
    return a + b
}

func main() {
    res := plus(2, 3)
    fmt.Println("res:", res)
}
```

## Multiple Return Values

---

```
package main

import "fmt"

func getPosition() (int, int) {
    return 3, 7
}

func main() {
    x, y := getPosition()
    fmt.Println("x:", x, "y:", y)
}
```

# Return Error



```
import "errors"

func dict(w string) (string, error) {
    m := map[string]string{"one": "หนึ่ง", "two": "สอง"}
    if v, hasWord := m[w]; hasWord {
        return v, nil
    }
    return "", errors.New("No this word")
}

func main() {
    s, err := dict("two")
    if err != nil {
        fmt.Println(err.Error())
        return
    }
    fmt.Println("s:", s)
}
```

# Closures (1)

---



```
package main

import "fmt"

func plus() func(a int, b int) int {
    return func(a int, b int) int {
        return a + b
    }
}

func main() {
    fn := plus()
    fmt.Println("plus:", fn(2, 3))
}
```

## Closures (2)

---



```
package main

import "fmt"

func main() {
    fn := func(a int, b int) int {
        return a + b
    }
    fmt.Println("plus:", fn(2, 3))
}
```

08

---

# Defer

```
package main

import "fmt"

func main() {
    defer fmt.Println("World")

    fmt.Println("Hello")
}
```



## Order of Defers

---

```
package main

import "fmt"

func main() {
    defer fmt.Println("1")
    defer fmt.Println("2")

    fmt.Println("3")
}
```

