

Getting Started with Google GoLang

...

Week 1



Why Learn ?

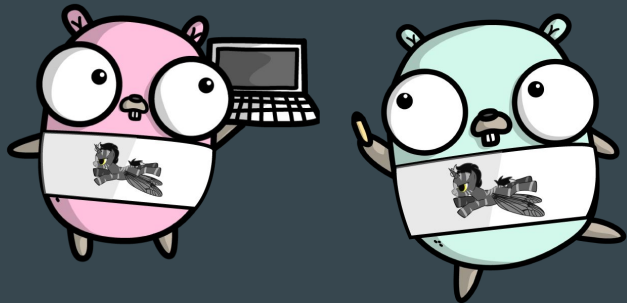
- Insanely fast
- Super Simple
- Extremely Powerful

Installing Go

Official Website: **golang.org**

Supports Linux, Windows and Mac.

Install guide: **golang.org/doc/install**



Code Editor Setup



Atom (**atom.io**)



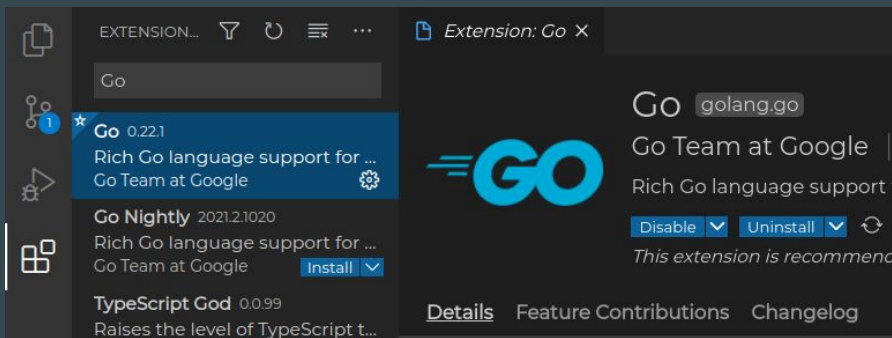
VS Code (**code.visualstudio.com**)



Vim (**vim.org**)

Go Plugins for VS Code

Simply search for Go in the extensions tab!









Hello World!

hello.go

```
1  package main
2
3  import "fmt"
4
5  func main() {
6      fmt.Println("Hello World!")
7  }
8  |
```

Execution and Output

```
 Week-1 > ls
hello.go
 Week-1 > go build hello.go
 Week-1 > ls
hello  hello.go
 Week-1 > ./hello
Hello World!
 Week-1 > go run hello.go
Hello World!
 Week-1 > |
```

Data types in Go

Go supports data types like int, float, strings, etc.

For more info about Go types, take a look at the following link:

[geeksforgeeks.org/data-types-in-go/](https://www.geeksforgeeks.org/data-types-in-go/)





Variables and Data types

data-types.go

```
1  package main
2
3  import "fmt"
4
5  func main() {
6      // Decleration followed by intitilisation
7      var x int
8      x = 0
9
10     // Decleration and intitilisation
11     var y float64 = 1.2
12
13     // Type inferring
14     z := "Hi!"
15
16     // Unused variable, not allowed
17     a := 123
18
19     fmt.Println("Int:", x)
20     fmt.Println("Float:", y)
21     fmt.Println("String:", z)
22 }
23
```

Execution and Output

```
 Week-1 > go run data-types.go  
Int: 0  
Float: 1.2  
String: Hi!  
 Week-1 > |
```

Composite types in Go

Go supports composite types like arrays, slices, maps, structs, etc.

For more info about composite types in Go, take a look at the following link:

tutorialedge.net/golang/go-complex-types-tutorial



Composite types - Arrays

arrays.go

```
1  package main
2
3  import "fmt"
4
5  func main() {
6      // Declaring an empty array
7      // var variableName [size]type
8      var arr [2]int
9
10     // Assigning values to array elements
11     arr[0] = 1
12     arr[1] = 2
13
14     fmt.Println("Array:", arr, "Length:", len(arr))
15
16     // Declaring and initialising an array
17     arr2 := [...]string{"Hello", "There"}
18
19     // In the above declaration ... is used to
20     // infer size, you can explicitly specify
21     // a number too. Eg: a := [3]int{1, 2, 3}
22
23     fmt.Println("Array 2:", arr2, "Length:", len(arr2))
24 }
25
```

Composite types - Slices

```
slices.go
1  package main
2
3  import "fmt"
4
5  func main() {
6      // Declaring an empty slice
7      // var variableName []type
8      var slice []int
9
10     fmt.Println("Slice before appends")
11     fmt.Println("Slice:", slice, "Length:", len(slice))
12
13     // Adding values to slices
14     // slice = append(slice, elements)
15     slice = append(slice, 1)
16     slice = append(slice, 2)
17
18     fmt.Println("Slice after appends")
19     fmt.Println("Slice:", slice, "Length:", len(slice))
20
21     // Declaring and initialising a slice
22     slice2 := []string{"Hello", "There", "General", "kenobi"}
23
24     // Deriving slice from other slice
25     // You can also derive slices from arrays
26     slice3 := slice2[1:3]
27
28     fmt.Println("Slice 2", slice2)
29     fmt.Println("Slice 3", slice3)
30 }
```

Other Composite types

More about Arrays and Slices:

- technobeans.com/2019/01/27/golang-composite-data-types-arrays-and-slices/

Structs and Maps:

- tutorialedge.net/golang/go-complex-types-tutorial
- [geeksforgeeks.org/structures-in-golang/](https://www.geeksforgeeks.org/structures-in-golang/)
- [geeksforgeeks.org/golang-maps/](https://www.geeksforgeeks.org/golang-maps/)



Conditionals: if-else

if.go

```
1  package main
2
3  import "fmt"
4
5  func main() {
6      x := 10
7      y := false
8      z := "hi!"
9
10     if x > 5 {
11         fmt.Println("X is greater than 5")
12     }
13
14     if x == 20 {
15         fmt.Println("20!")
16     } else if x < 20 && !y {
17         fmt.Println("< 20")
18     } else {
19         fmt.Println(z)
20     }
21 }
22
```

Loops

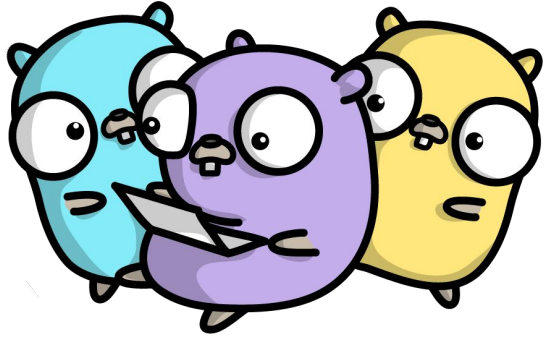
loops.go

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     arr := []int{}
7
8     // 3 condition for loop
9     // for <initialisation>; <condition>; <update>
10    for i := 0; i < 10; i++ {
11        arr = append(arr, i*i)
12    }
13
14    y := len(arr) - 1
15
16    // While loop
17    // for <condition>
18    for y >= 0 {
19        arr[y]++
20        y--
21    }
22
23    // Infinite loop
24    for {
25        fmt.Println("Infinite!")
26    }
27 }
28
```


Iterating over arrays

arr-loop.go

```
1  package main
2
3  import "fmt"
4
5  func main() {
6      arr := []int{1, 2, 3, 4, 5, 6}
7
8      // Iterating over a slice
9      for index, value := range arr {
10         fmt.Println(index, value)
11     }
12
13     sum := 0
14     // Use _ when we don't require a variable
15     for _, value := range arr {
16         sum += value
17     }
18
19     fmt.Println(sum)
20 }
21
```



Thank You!

Source Code and Slides available at:
github.com/Gituser143/PESU-IO-Go