GoLang

* Founder :  [Robert Griesemer](https://en.wikipedia.org/wiki/Robert_Griesemer), [Rob Pike](https://en.wikipedia.org/wiki/Rob_Pike), and [Ken Thompson](https://en.wikipedia.org/wiki/Ken_Thompson)
* Developed by Google in 2007.
* Open Sourced in 2009.
* Go is Statically typed, Compiled high-level Programming language.

**Note : I’m creating these notes as considering the reader have teachnical background. Reader have a basic knowledge of any other programming language.**

1. What is Golang used for ?

* Go is popular in Cloud-based application or Server-side app.
* DevOps & website reliability automation are two well know ways to utilize Go.
* A lot of command-line tools have been written in Go.
* Go is used in AI & Data Science.

0.1 Advantage

* Fast
* Easy to learn
* Well scaled
* Comprehensive Programming Tool
* Strong Community Support (Google)

0.2 Disadvantage

* Young Language

0.3 Features

* Light weight (Goroutine take 8kb)
* Simplicity
* Concurrency
* Garbage Collection
* Cross Platform Support
* Fast Compile Time
* Strong Typing
  1. Installation of GoLang
* Go to <https://go.dev/doc/install> & download the installer according to Operating System.
* After installation run the command **go version** on command prompt for checking the successful installation of Go.

0.5 Default Directory Structure

* After installation it creates **go** directory in our **Root** directory.
* To know the path of directory run **go env GOPATH** command.
* In my case it shows **C:\Users\Prajwal\go** & this path also known as **GOPATH**.
* Following is a Default structure of Directory :

go/

bin/

( executable binaries )

pkg/

( compiled package file )

src/

( Myproject )

* In starting there are only 2 folders are present in directory first bin & second pkg, src folder is created by user.
* According to go standards it says do all coding under Go Workspace means inside go directory.
* But according to our convenient we can create our project folder outside GOPATH, but we have to perform some extra steps.

0.6 Go Module

* Go module helps to communicate with GOPATH, it imports the required packages from GOPATH.
* Go Module simplify the management of dependencies & project structure.
* Go Module allows you to create & manage the projects outside the GOPATH.
* To initialize the folder as a Module we have to run **go mod init MODULE\_NAME** command at our folder path in terminal.
* After running command it creates **go.mod** file.
* go.mod file contains the information about our project, including its name & dependencies.
* This process is mandatory when we create our project or folder out side the GOPATH.

0.7 First Program

package main

import “fmt”

func main(){

fmt.println(“Hello World!!!”)

}

0.8 Packages

* Go use Packages instead of classes.
* Each go file must belong to some package.
* Syntax to define package :

**package PACKAGE\_NAME**

* The **main** package is a special package in go. An executable program must contain the **main** package.
* Go uses relative imports to bring packages into current file.
* We can import packages using **import** keyword.
* **main** function is an entry point of our executable program. It should be under the main package.

0.9 Variables

* With the help of **var** & **const** keyword we can initialize or declare the variables.
* There are few ways to create variables.

Case 1 :

var variable\_name data\_type = value

var str1 string = “String 1”

Case 2 :

var variable\_name

var num

Case 3 :

variable\_name := value

pi := 3.14

* In case 1 we specify the datatype of variable. We can’t change datatype of variable later.
* In case 2 we only specify the variable name, we can change the datatype of variable according to our data.
* In case 3 we use **:=** operator instead of var keyword. According to data it will initialize the datatype to variable.
* We can create Constant variables with const keyword & normal variables with var.
* We can’t change the value after declaring the constant variable.
* Ex :

Const pi = 3.14

* When we want to export any variable or function then its first letter of variable must be **capital**. We can export it in various packages & files.
* If variable name is in lowercase then that variable is accessible in only that file. We can’t export it for external use.
* Ex :

var Public\_variable

var private\_variable

**Questions**

1. What is the syntax for declaring a variable in Go?
2. How do you declare multiple variables in a single line in Go?
3. What is the zero value in Go, and how does it relate to variables?
4. How do you declare a constant in Go?
5. What is the short variable declaration, and when should it be used?
6. How do you swap values of two variables in Go?
7. Can you reassign a value to a variable declared with the := syntax? Explain.
8. What is the scope of a variable in Go?
9. How do you declare a variable without assigning an initial value?
10. What happens if you try to use a variable that has not been initialized?

1.0 Input & Output

* We can write the output with three methods of **fmt** package.
* Println :
  + It prints the statement and add space before the variable printing.
  + After printing the statement set the cursor on next line.
* Print :
  + It can’t add space & can’t set the cursor on next line.
  + It only print the statements.
* Printf :
  + It works like printf() function of C programming.
  + It use format specifiers for printing the statements.
* There are some several ways to take input according to user data.
* There are three **fmt** package methods define for taking input.
* Scan :
  + It is use to take single value at a time.
  + Scan scans text read from standard input, storing successive space-separated values into successive arguments.(referred form website)
* Scanln :
  + Scanln is similar to Scan, but stops scanning at a newline and after the final item there must be a newline or EOF.
* Scanf :
  + It is similar to Scanf() function of C Programming.
  + It use format specifier to take input according to data.
* These functions are take only one word string. We want to use BufferReader for accepting long string.
* We can take non string data easily with these functions.

**Questions**

1. How do you print output to the console in Go?

2. How do you read input from the console in Go?

3. What is the fmt package, and what are its common functions for I/O operations?

4. How do you format strings using the fmt package?

5. What is the difference between fmt.Print, fmt.Println, and fmt.Printf?

6. How do you handle errors when reading input in Go?

7. How do you read an entire line of input from the console?

* 1. Functions
* Functions are declared with **func** keyword, followed by name, parameter (if any), return type or variable (if any) and the function body.
* Syntax :

func Function\_name(Parameters) (variable & Return\_type){

//body

}

* Examples :

// function without parameter & return type

func f1(){

fmt.Println(“Function 1”)

}

// function with parameter & return type

func f2(n1, n2 int) int {

return n1+n2

}

// function with parameter & variable return type

func f3(n1, n2 int) (ans int) {

ans= n1+n2

return

}

* We can give common datatype in input parameter if there is same datatype variable. We can give it separately also.

func f2(n1 int, n2 int)

* In example 3, we declare the return variable & return type of function. **ans** is return variable & **int** is return type. If we declare the return variable then it is not necessary to mention it again in return statement, it will automatically return the value of return variable i.e. **ans**.
* Opening curly bracket **{** must be on same line of function declaration. It is declared structure of GoLang.

**Questions**

 **Function Declaration:**

* How do you declare a function in Go?
* What is the syntax for defining a function that returns a value in Go?

 **Multiple Return Values:**

* How can a function return multiple values in Go?
* Provide an example of a function that returns two integers.

 **Named Return Values:**

* What are named return values, and how are they used in Go functions?
* Give an example of a function using named return values.

 **Variadic Functions:**

* What is a variadic function in Go?
* How do you define and call a variadic function?

 **Function as a Value:**

* How can you assign a function to a variable in Go?
* Provide an example where a function is passed as an argument to another function.

 **Anonymous Functions:**

* What is an anonymous function in Go?
* How do you define and invoke an anonymous function?

 **Closures:**

* What is a closure in Go?
* Provide an example of a closure in Go.

 **Defer Statement:**

* What is the defer statement, and how is it used in functions?
* Give an example demonstrating the use of defer in a function.

 **Recursive Functions:**

* What is recursion, and how do you write a recursive function in Go?
* Provide an example of a recursive function.

 **Method vs. Function:**

* What is the difference between a method and a function in Go?
* How do you define a method on a type in Go?