Go Lang Foundation Course - Beginner to Intermediate

Course Description & Objectives:

Go is an open source programming language created by Google. As one of the fastest growing languages in terms of popularity, its a great time to pick up the basics of Go!

Will be able to program in Golang, understand and review code written by others, have foundation to build on and work with different libraries to write any program.

Duration:

3 Days

Pre-requisites:

Should have programmed in another language like C, C++, Java, C#, Python, etc. Should have written programs using control structures.

Software & Hardware requirements:

Internet connectivity, Laptop with Windows/Linux/MacOS

High Level Course Contents

- The Go Language, Control Structures
- "OOP" - Structs, Interfaces, Encapsulation, Inheritance, Polymorphism
- Concurrency
- Web Programming

Day wise coverage:

Course Outline:

Module 1: The Go Language, Control Structures

(up to noon of day 1)

Why Go?

- The Beginnings of Go
- Philosophies of Go
- Go vs Other languages
- Supported Platforms, Cross Compiling
- Key Distinguishing Features

Setting Up Go

- Downloading and Installing Go

- Setting up Go Environment Variables
- Why do we need Git, Mercurial, etc.?
- Go Playground

Basic Program, Go Tools

- Hello World, packages, import, main & main
- Go run

Working with Strings

- String Functions
- String Formatting

Variables and Assignment

- var, := differences
- new
- Multiple assignment
- Values
- Variables
- Constants

Defer Statement

- Defer

Functions

- Writing a Function
- Named Return Values
- Multiple Return Values

Errors

- Errors in Go
- panic and recover

Pointers, Parameters, Return Values

- Pointers
- Pass by Value, Pass by Reference

Control Statements

- If/Else

- Switch

Arrays, Slices, Maps, for

- For
- Arrays
- Slices
- Maps
- Range

defer, continue, break, goto, fallthrough

- continue, break, goto with label
- fallthrough for switch

Module 2: "OOP" - Structs, Interfaces, Encapsulation, Inheritance, Polymorphism

(noon of day 1 to noon of day 2)

More About Functions

- Variadic Functions
- Closures

Structs

- Structs
- Struct Members
- Anonymous Struct Members
- Methods on Structs
- Pointer and Value Receivables in struct methods
- How structs take place of objects

Interfaces

- Interfaces
- Implicitness of Interfaces
- Example with io.Reader and io.Writer

Encapsulation, "Object Hierarchy", "OOP"

- Data Hiding
- Struct Inheritance with Composition

Polymorphism

Module 3: Concurrency

(noon of 2nd day to noon of 3rd day)

Attendee pre-requisites: Module 1 and Module 2.
Notes : This module is fundamental to Go and very different from common programming languages – though it sounds like threads, it is not, and the coordination of concurrency is vital to learn. This module is a must and is also one of the most powerful aspects of Go which makes it very efficient and fast.

Concurrency: Goroutines, Parallelism
- Concurrency with goroutines
- Concurrency and Parallelism

Timers and Tickers
- Tickers
- Timeouts
- Timers
- Epoch

Concurrency: SyncGroup, Wait, Mutexes
- Sync, Wait
- Mutexes
- Deadlocks with Mutexes
- RW Mutexes

Concurrency: Handling Race Conditions
- Example of Race Condition

Concurrency: Channels
- Channels
- Channel Direction
- Closing Channels
- Range Over Channels
- Channels - Select
- Timeouts
- extra: Sending and Receiving on Closed Channels
- extra: Select on Closed Channels

Concurrency: Miscellaneous
- Work Stealing
- Goroutines are Independent of Each Othe

Concurrency: Canceling and Context
- Leaky Goroutines
- Canceling Goroutines
- Canceling with Context
- Canceling with Deadline/Timeout

Closing a Context Sends Done() to All
Closing a Context Sends Done() to All Derived Contexts

Module 4: Web Programming

JSON and XML

- Struct to JSON
- Marshaling and Unmarshaling
- JSON Tags
- Working with XML

http package

- Running a Web Server and Handling Requests
- HTTP Return Codes
- Serving Static Files
- Context

gorilla package

- Installing gorilla mux
- Routing URLs
- Sub routers