Go Programming Language

Beginner To Advanced

Trainer
Amarjit Singh

Trainer
Amarjit Singh

Course : Go Programming Language
Level : Beginner to Advanced
Duration : 12 Full Days [Total 78 Hours] ≈ 20 Half Days *
[4 Hours/Half Day Session]

**** NOTE:***

8 Hours/Full Day Training session duration consists of 1 hour lunch break, 15 minutes two tea/coffee breaks. Hence ***Effective Training Hours Per Full Day will be 6 and ½ Hours*** *after excluding breaks from training session duration.*

Objectives

Upon completion of this course, trainees will be able to:

- Programming in Go Language
- Go Language Objectives and Design
- Understanding the object-oriented features of Go
- Understanding of Important Functions and Libraries in Go
- Go Concurrency Model
- Go Concurrent Programming
- Transitioning from C to Go Language

Audience

This course is designed for developers interested in learning Go Programming at intermediate to advanced levels.

Prerequisites

Participants *must* know

Concepts of Object Oriented Programming/Paradigm
Programming in C/C++ Language
Energy and Openness to Learn, Attitude to Take Ownership
Appreciation and Openness to Design Thinking
Understanding and Appreciation of Mathematics 10+2 Level
Strong Logical and Programming Ability
Following Training Schedule and Decorum
Arriving on Time and Avoiding Distractions

Hardware/Software Requirements

For Go Programming:
System: Core i5+ Microprocessor with 8GB+ RAM
Operating System: Windows 10+ or Ubuntu Desktop 22.04 LTS 64 Bit or MacOSX
IDE/Compiler: Go Compiler and Sublime Text/Visual Studio Code/Vim or Any Source Code Editor
Others: Projector/Whiteboard/Marker/Duster and Internet

Go Programming Language

Program Structure *

This course is divided into following courses with projected days distribution.

ESSENTIAL SECTIONS

SECTIONS	TITLE	DURATION *
COURSE A1 **COURSE A2** **COURSE A3**	**Go Programming Language** **Go Concurrent Programming** **Go Web Programming**	**06 Days** **02 Days** **04 Days** _____ **Total 12 Full Days = 78 Total Hours** **= 78 Effective Training Hours** **= 78 Hours / 4 Hours For Half Day** **≈ 20 Half Days**

**** NOTE:***

1. ***8 Hours/Full Day Training session duration consists of 1 hour lunch break, 15 minutes two tea/coffee breaks. Hence Effective Training Hours Per Full Day will be 6 and ½ Hours after excluding lunch and tea/coffee breaks from Full Day training session duration.***

2. *Depth and breadth of Modules/Sections covered depends on participants and program level, objectives, duration of the course, participants diversity, prior knowledge and participants meeting the prerequisites and total time available during the training.*

3. *This training is designed for beginner to advanced level. Trainer will prioritise topics to reach around 90%+ coverage on best effort basis depending on participants diversity, prior knowledge and unlearning/learning/grasping capabilities, time available during the training etc. Optional marked modules will be covered if time permits.*

4. *Generally on the average we cover 2 Modules per Full Day session.* ***Please note that fitting more numbers of Modules per Full Day session, can lead to reduction in interactions, hand on time %, time for queries/questions, discussions and in depth coverage. In this scenario the trainer will try to complete topics by prioritising on the best effort basis and require participants cooperation in terms of taking ownership in this trainer guided learning and following the guidelines and instructions given by the trainer.***

Training Pedagogy, Methodology and Conventions *

Training pedagogy, methodology, mechanisms and training techniques are decided by the trainer based on many factors including duration, agenda, objectives, topics, trainees diversity, meeting prerequisites etc. He will be the sole decision maker in terms of, how topics to be covered and prioritisation of topics. Training objectives are to build competency of trainees in required know-how to face programming challenges. Solutions building/specific problems solving are generally part of consultancy rather than training, Specific examples are covered as proof of concepts rather than full solutions. Please note this training is not designed for any specific learner, rather for class as a whole.

Go Programming Language

Detailed Course Outline

SECTION A1: GO PROGRAMMING LANGUAGE

MODULE 01: INTRODUCTION TO GO

- Solving modern programming challenges with Go
  - Development Speed
  - Concurrency
  - Go's type system
  - Memory management
- Go Playground
  - Getting Started with Go

MODULE 02: PROGRAM STRUCTURE

- Names
- Declarations
- Variables
- Assignments
- Type Declarations
- Packages and Files Scope
- Main package
- Search package
  - search.go
  - feed.go
  - match.go/default.go
- Program Architecture

MODULE 03: GO TYPE SYSTEM

GO BASIC DATA TYPES

- Integers
- Floating-Point Numbers
- Complex Numbers
- Booleans
- Strings
- Constants

GO TYPE SYSTEM

- Methods
- The nature of types
  - Built-in types
  - Reference types
  - Struct types
- Interfaces
  - Standard library
  - Implementation
  - Method sets
  - Polymorphism
- Type Embedding

Exporting and unexporting identifiers

MODULE 04: GO COMPOSITE TYPES

ARRAYS, SLICES AND MAPS

Array internals and fundamentals
- Internals
- Declaring and initialising
- Working with arrays
- Multidimensional arrays
- Passing arrays between functions

Slice internals and fundamentals
- Internals
- Creating and initialising
- Working with slices
- Multidimensional slices
- Passing slices between functions

Map internals and fundamentals
- Internals
- Creating and initialising
- Working with maps
- Passing maps between functions

STRUCTS AND JSON

Structs
JSON
Text and HTML Templates

MODULE 05: GO FUNCTIONS

Function Declarations
Recursion
Multiple Return Values
Errors
Function Values
Anonymous Functions
Variadic Functions
Deferred Function Calls
Panic
Recover

MODULE 06: GO METHODS

Method Declarations
Methods with a Pointer Receiver
Composing Types by Struct Embedding
Method Values and Expressions
Encapsulation

MODULE 07: GO INTERFACES

Interfaces as Contracts
Interface Types
Interface Satisfaction
Parsing Flags with flag.Value

Interface Values
Sorting with sort.Interface
The http.Handler Interface
The error Interface
Expression Evaluator
Type Assertions
Discriminating Errors with Type Assertions
Querying Behaviours with Interface Type Assertions
Type Switches
Best Practices

MODULE 08: GO STANDARD LIBRARY

Documentation and Source Code
Logging
- Log Package
- Customised Loggers
- Conclusion

Encoding/Decoding
- Decoding JSON
- Encoding JSON
- Conclusion

Input and Output
- Writer and Reader Interfaces
- Working Together
- Simple Curl

MODULE 09: GO CONCURRENCY OVERVIEW

CONCURRENCY Vs PARALLELISM
- Goroutines
- Race conditions
- Locking shared resources
  - Atomic functions
  - Mutexes
- Channels
  - Unbuffered channels
  - Buffered channels
- Concurrent Traversal
- Cancellation

CONCURRENCY WITH SHARED VARIABLES
- Race Conditions
- Mutual Exclusion: sync.Mutex
- Read/WriteMutexes: sync.RWMutex
- Memory Synchronisation
- Lazy Initialization: sync.Once

MODULE 10: PACKAGING AND TOOLING

Packages
- The Package Declaration
- Package-naming conventions
- Package main

Imports

- Import Declarations
- Import Paths
- Remote imports
- Named imports
- Blank Imports
- init
- Using Go tools
- Going further with Go developer tools
  - Go vet
  - Go format
  - Go documentation
- Collaborating with other Go developers
  - Creating repositories for sharing
- Dependency management
  - Vendoring dependencies
  - Introducing gb

MODULE 11: REFLECTION IN GO

- Why Reflection?
- reflect.Type and reflect.Value
- Display, a Recursive Value Printer
- Setting Variables with reflect.Value
- Accessing Struct Field Tags
- Displaying the Methods of a Type
- Best Practices

MODULE 12: LOW LEVEL PROGRAMMING

- unsafe.Sizeof, Alignof, and Offsetof
- unsafe.Pointer
- Calling C Code with cgo
- Best Practices

MODULE 13: TESTING AND BENCHMARKING

- Unit testing
  - Basic unit test
  - Table tests
  - Mocking calls
  - Testing endpoints
- Benchmarking

SECTION A2: GO CONCURRENT PROGRAMMING

MODULE 01: AN INTRODUCTION TO CONCURRENCY

- Moore’s Law, Web Scale, and the Mess We’re In
- Why Is Concurrency Hard?
  - Race Conditions
  - Atomicity
  - Memory Access Synchronisation
  - Deadlocks, Livelocks, and Starvation
  - Determining Concurrency Safety
  - Simplicity in the Face of Complexity

MODULE 02: COMMUNICATING SEQUENTIAL PROCESSES

- The Difference Between Concurrency and Parallelism
- What Is CSP?
- How This Helps You
- Go's Philosophy on Concurrency

MODULE 03: GO'S CONCURRENCY BUILDING BLOCKS

- Goroutines
- The sync Package
  - WaitGroup
  - Mutex and RWMutex
  - Cond
  - Once
  - Pool
- Channels
- The select Statement
- The GOMAXPROCS Lever

MODULE 04: CONCURRENCY PATTERNS IN GO

- Confinement
- The for-select Loop
- Preventing Goroutine Leaks
- The or-channel
- Error Handling
- Pipelines
  - Best Practices for Constructing Pipelines
  - Some Handy Generators
- Fan-Out, Fan-In
- The or-done-channel
- The tee-channel
- The bridge-channel
- Queuing
- The context Package

MODULE 05: CONCURRENCY AT SCALE

- Error Propagation
- Timeouts and Cancellation
- Heartbeats
- Replicated Requests
- Rate Limiting
- Healing Unhealthy Goroutines

MODULE 06: GOROUTINES AND THE GO RUNTIME

- Work Stealing
  - Stealing Tasks
  - Continuations
- Designing Code

MODULE 07: GOROUTINES DEBUGGING

- Anatomy of a Goroutine Error
- Race Detection
- pprof

SECTION A3: GO WEB PROGRAMMING

MODULE 01: WEB APPLICATIONS INTRODUCTION

- Using Go for web applications
  - Scalable web applications and Go
  - Modular web applications and Go
  - Maintainable web applications and Go
  - High performing web applications and Go
- How web applications work
- A quick introduction to HTTP
- The coming of web applications
- HTTP request
  - Request methods
  - Safe request methods
  - Idempotent request methods
  - Browser support for request methods
  - Request headers
- HTTP response
  - Response status code
  - Response headers
- URI
- Introducing HTTP/2
- Parts of a web app
  - Handler
  - Template engine
- Hello Go

MODULE 02: GO FIRST WEB APPLICATION

- Application design
- Data model
- Receiving and processing requests
  - The multiplexer
  - Serving static files
  - Creating the handler function
  - Access control using cookies
- Generating HTML responses with templates
  - Tidying up
- Installing PostgreSQL
  - Linux/FreeBSD
  - Mac OS X
  - Windows
- Interfacing with the database
- Starting the server
- Wrapping up

MODULE 03: HANDLING REQUESTS

- The Go net/http library
- Serving Go
  - The Go web server
  - Serving through HTTPS
- Handlers and handler functions
  - Handling requests
  - More handlers
  - Handler functions

- Chaining handlers and handler functions
- ServeMux and DefaultServeMux
- Other multiplexers
- Using HTTP/2

MODULE 04: PROCESSING REQUESTS

- Requests and responses
  - Request
  - Request URL
  - Request header
  - Request body
- HTML forms and Go
  - Form
  - PostForm
  - MultipartForm
  - Files
  - Processing POST requests with JSON body
- ResponseWriter
  - Writing to the ResponseWriter
- Cookies
  - Cookies with Go
  - Sending cookies to the browser
  - Getting cookies from the browser
  - Using cookies for flash messages

MODULE 05: DISPLAYING CONTENT

- Templates and template engines
- The Go template engine
  - Parsing templates
  - Executing templates
- Actions
  - Conditional actions
  - Iterator actions
  - Set actions
  - Include actions
- Arguments, variables, and pipelines
- Functions
- Context awareness
  - Defending against XSS attacks
  - Unescaping HTML
- Nesting templates
- Using the block action to define default templates

MODULE 06: STORING DATA

- In-memory storage
- File storage
  - Reading and writing CSV files
  - The gob package
- Go and SQL
  - Setting up the database
  - Connecting to the database
  - Creating a post
  - Retrieving a post
  - Updating a post

- Deleting a post
- Getting all posts
- Go and SQL relationships
  - Setting up the databases
  - One-to-many relationship
- Go relational mappers
  - Sqlx
  - Gorm

MODULE 07: GO WEB SERVICES

- Introducing web services
- Introducing REST-based web services
  - Convert action to a resource
  - Make the action a property of the resource
- Parsing and creating XML with Go
  - Parsing XML
  - Creating XML
- Parsing and creating JSON with Go
  - Parsing JSON
  - Creating JSON
- Creating Go web services

MODULE 08: LEVERAGING GO CONCURRENCY

- Concurrency isn't parallelism
- Goroutines
  - Using goroutines
  - Goroutines and performance
  - Waiting for goroutines
- Channels
  - Synchronisation with channels
  - Message passing with channels
  - Buffered channels
  - Selecting channels
- Concurrency for web applications
  - Creating the photo mosaic
  - The photo mosaic web application
  - Concurrent photo mosaic web application

MODULE 09: TESTING YOUR APPLICATION

- Go and testing
- Unit testing with Go
  - Skipping test cases
  - Running tests in parallel
  - Benchmarking
- HTTP testing with Go
- Test doubles and dependency injection
  - Dependency injection with Go
- Third-party Go testing libraries
  - Introducing the gocheck testing package
  - Introducing the Ginkgo testing framework

MODULE 10: DEPLOYING GO WEB APPLICATION

- Deploying to servers
- Deploying to Heroku