ISInc
Innovative Solutions

# GO Language Training

Course ISI-1417  4 Days - Instructor-led, Hands on

## Introduction

This course provides a solid foundation for GO language programming using all the features the language provides, as well as the most commonly used GO packages. In this course, GO's unique features and idioms are illustrated with complete runnable examples.

For years—decades, really—there have been less than a handful of options for writing servers and network interfaces. If you were tasked with writing one, you probably used C, C++, or Java. And while these certainly can handle the task, and while they all now support concurrency and parallelism in some way or another, they weren't designed for that.

GOogle brought together a team that included some giants of programming—Rob Pike and Ken Thompson of Bell Labs fame and Robert Griesemer, who worked on GOogle's JavaScript implementation V8—to design a modern, concurrent language with development ease at the forefront.

GO runs on both Windows and Unix-like operating systems such as Linux and Mac OS X. Arguably, C++, Objective-C, and Java (the latter indirectly as a "better C++"), have all attempted to be better C's. GO is closer in spirit to C than to any other language, and can be seen as an attempt to avoid all of C's drawbacks while providing all that's best in C, as well as its own powerful and useful features.

GO provides high level features such as slices (in effect variable length arrays) and maps (hash tables). GO can be used for procedural and object-oriented programming (or a combination of both). GO's approach to object orientation is unusual being based on aggregation and delegation, interfaces, and duck typing, rather than inheritance and virtual (overridable) methods. GO has a garbage collector to relieve programmers from the burdens of manual memory management. Built into the GO language is support for a version of CSP (Concurrent Sequential Processes) using channels that makes writing concurrent programs a lot easier than the traditional threading approaches—it is perfectly possible to write highly concurrent GO programs that contain no explicit locks at all.

The course will be useful to people who program professionally as part of their job, whether as full-time software developers, or those from other disciplines, including scientists and engineers, who need to do some programming in support of their work. It will also be useful to students who have had a reasonable amount of programming experience.

ISInc
Innovative Solutions

## Prerequisites

Programming experience in a mainstream programming language such as C, C++, Java, Python, or similar.

## Course Objectives

Students will learn to:

- Create GO applications using the LiteIDE and Sublime
- Create GO applications that use Booleans and numbers
- Create GO applications that use strings
- Create GO applications that use collection types
- Create GO applications with procedural programming
- Create GO applications with object oriented programming
- Create GO applications with simple concurrent programming
- Create GO applications that incorporate file handling
- Create GO packages
- Create GO applications with advanced concurrency features that include: goroutines and channels, and will compare the way GO handles concurrency with the approach other languages use
- Create GO applications that include resource allocation, sharing memory (and when not to), and data
- Design GO applications to best use concurrent tools in GO
- Create GO applications that ensure the delegation of goroutines and channels maintain the state in single thread and multithread applications
- Create GO applications that avoid dead locks out of the box, and when and where they can still occur despite GO's language design
- Build a Non-blocking Web Server in GO
- Address Performance and Scalability in a GO application
- Describe a Concurrent Application Architecture in GO and when and where to implement concurrent patterns, when and how to utilize parallelism to take advantage of advanced hardware, and how to ensure data consistency
- Create GO applications that incorporate Logging and Testing
- Implement more complicated and advanced techniques including duplicating concurrent features not available in GO's core.

## Course Outline

### Module 1: An Overview in Five Examples

- Getting GOing
- Editing, Compiling, and Running
- Hello Who?
- Big Digits–Two-Dimensional Slices
- Stack–Custom Types with Methods
- Americanise–Files, Maps, and Closures
- Polar to Cartesian–Concurrency
- Lab

### Module 2: Booleans and Numbers

- Preliminaries

- Boolean Values and Expressions
- Numeric Types
- Example: Statistics
- Lab

### Module 3: Strings

- Literals, Operators, and Escapes
- Comparing Strings
- Characters and Strings
- Indexing and Slicing Strings
- String Formatting with the Fmt Package
- Other String-Related Packages
- Example: M3u2pls
- Lab

### Module 4: Collection Types

- Values, Pointers, and Reference Types
- Arrays and Slices
- Maps
- Examples
- Lab

### Module 5: Procedural Programming

- Statement Basics
- Branching
- Looping with For Statements
- Communication and Concurrency Statements
- Defer, Panic, and Recover
- Custom Functions
- Example: Indent Sort
- Lab

### Module 6: Object-Oriented Programming

- Key Concepts
- Custom Types
- Interfaces
- Structs
- Examples
- Lab

### Module 7: Concurrent Programming

- Key Concepts
- Examples
- Lab

### Module 8: File Handling

- Custom Data Files

- Archive Files
- Lab

## Module 9: Packages

- Custom Packages
- Third-Party Packages
- Brief Survey of GO's Commands
- A Brief Survey of the GO Standard Library
- Lab

## Module 10: An Introduction to Concurrency in GO

- Introducing goroutines
- Implementing the defer control mechanism
- Understanding goroutines versus coroutines
- Implementing channels
- Closures and goroutines
- Building a web spider using goroutines

## Module 11: Understanding the Concurrency Model

- Understanding the working of goroutines
- Synchronous versus asynchronous goroutines
- Visualizing concurrency
- RSS in action
- A little bit about CSP
- GO and the actor model
- Object orientation
- Using concurrency
- Managing threads
- Using sync and mutexes to lock data

## Module 12: Developing a Concurrent Strategy

- Applying efficiency in complex concurrency
- Identifying race conditions with race detection
- Synchronizing our concurrent operations
- The project – multiuser appointment calendar
- A multiuser Appointments Calendar
- A note on style
- A note on immutability

## Module 13: Data Integrity in an Application

- Getting deeper with mutexes and sync
- The cost of goroutines
- Working with files
- Getting low – implementing C
- Distributed GO
- Some common consistency models
- Using memcached

ISInc
Innovative Solutions

## Module 14: Locks, Blocks, and Better Channels

- Understanding blocking methods in GO
- Cleaning up goroutines
- Creating channels of channels
- Pprof – yet another awesome tool
- Handling deadlocks and errors

## Module 15: C10K – A Non-blocking Web Server in GO

- Attacking the C10K problem
- Building our C10K web server
- Serving pages
- Multithreading and leveraging multiple cores
- Exploring our web server
- Module 7: Performance and Scalability
- High performance in GO
- Using the App Engine
- Distributed GO
- Some helpful libraries
- Memory preservation

## Module 16: Concurrent Application Architecture

- Designing our concurrent application
- Identifying our requirements
- Using NoSQL as a data store in GO
- Monitoring filesystem changes
- Managing logfiles
- Handling configuration files
- Detecting file changes
- Backing up our files
- Designing our web interface
- Reverting a file's history – command line
- Checking the health of our server

## Module 17: Logging and Testing Concurrency in GO

- Handling errors and logging
- Using the log4go package for robust logging
- Using the runtime package for granular

## Module 18: Advanced Concurrency and Best Practices

- GOing beyond the basics with channels
- Building workers
- Implementing nil channel blocks
- Implementing more granular control over goroutines with tomb
- Timing out with channels
- Building a load balancer with concurrent patterns
- Choosing unidirectional and bidirectional channels
- Using an indeterminate channel type
- Using GO with unit testing

ISInc
Innovative Solutions

- Using GOogle App Engine
- Utilizing best practices