Go Programming (Go Lang)

Course Duration: 40 Hours (5 Days)

Overview

The Go Programming (Go Lang) course is designed to provide a comprehensive introduction to Go, an open-source programming language created by Google. This course covers everything from the basic structure and syntax of the language to its advanced features, such as Concurrency with goroutines and channels. With a focus on practical application, the course is structured into modules that guide learners through the Go environment setup, Basic syntax, Data types, Variables, Constants, Operators, decision making, Loops, Functions, Scope rules, Strings, Arrays, Pointers, Structures, Slices, Maps, Recursion, Type casting, Interfaces, Error handling, and Packages. By enrolling in the Golang Training, students will gain a solid foundation in Go Language Course principles and be able to build efficient and reliable software. The course is designed for both beginners and experienced programmers, offering a step-by-step approach to mastering Go. The hands-on lessons and examples will help learners to become proficient Go developers, ready to tackle real-world programming challenges.

Audience Profile

The Go Programming course by Koenig Solutions is tailored for developers aiming to master Google's Go language for building efficient and concurrent applications.

- Software Developers focused on backend system development.
- Programmers seeking to learn a modern, efficient programming language.
- DevOps Engineers interested in creating scripts and automation tools using Go.
- Computer Science Students specializing in systems programming.
- IT Professionals developing network servers or distributed systems.
- Technical Leads managing teams utilizing Go.
- Full Stack Developers expanding their expertise in backend development.
- System Architects designs scalable and high-performance applications.
- Cloud Computing Experts working on Go-supported infrastructure.
- Open-Source Contributors contributing to projects written in Go.
- Mobile App Developers seeking to understand Go for server-side systems.
- Data Scientists requiring performance-optimized code for data processing tasks.

Course Syllabus

Day 1: Introduction to Go Programming

- Objective: Gain familiarity with Go language basics, environment setup, and writing first programs.

KOENIG
step forward

Morning Session (Theory)

1. **Overview of the Go Programming Language**
    a. History and key features of Go
    b. Why Go? Use cases and applications
2. **Installing Go & Development Environment**
    a. Setting up the Go workspace
    b. Configuring VS Code/GoLand for Go development
3. **Basic Syntax and Structure**
    a. Packages, imports, and functions
    b. Variables, constants, and data types
    c. Basic input/output operations

Afternoon Session (Lab)

1. **Lab 1: Go Installation & Hello World**
    a. Install Go and set up the IDE
    b. Create a Go workspace
    c. Write, compile, and run a "Hello, World" program
2. **Lab 2: Variables and Functions Practice**
    a. Create variables of different types
    b. Write functions and pass parameters

Day 2: Control Structures, Arrays, and Slices

- Objective: Understand control flow and core data structures.

Morning Session (Theory)

1. **Control Structures**
    a. If-else and switch statements
    b. Loops: for, range, and looping structures
2. **Arrays and Slices**
    a. Declaring, initializing, and iterating over arrays
    b. Introduction to slices (dynamic arrays)
    c. Manipulating slices (append, copy, slice)

Afternoon Session (Lab)

1. **Lab 3: Working with Control Structures**
    a. Implement a program using if, switch, and for statements
    b. Build a number guessing game using loops
2. **Lab 4: Manipulating Arrays and Slices**

a. Perform operations such as adding, deleting, and updating elements in slices

Day 3: Go Functions, Pointers, and Error Handling

- Objective: Explore functions, pointers, and Go's approach to error handling.

Morning Session (Theory)

1. **Functions in Go**
   a. Function signatures and return types
   b. Passing by value vs. passing by reference
   c. Variadic functions
2. **Pointers**
   a. Introduction to pointers and dereferencing
   b. Working with pointer types in Go
3. **Error Handling**
   a. Error handling the Go way (multiple return values)
   b. Custom error types and the errors package
   c. Panic and recover mechanisms

Afternoon Session (Lab)

1. **Lab 5: Function Creation and Usage**
   a. Create various functions, including variadic functions
   b. Work with function return types and multiple return values
2. **Lab 6: Pointers and Error Handling**
   a. Use pointers to modify values
   b. Handle errors in a sample program and recover from panics

Day 4: Structs, Methods, and Interfaces

- Objective: Explore Go's object-oriented features using structs, methods, and interfaces.

Morning Session (Theory)

1. **Structs**
   a. Defining and using structs
   b. Nested structs
   c. Struct methods and receivers
2. **Interfaces**
   a. Introduction to Go interfaces
   b. Implementing and using interfaces
   c. Empty interfaces and type assertions

KOENIG
step forward

Afternoon Session (Lab)

1. **Lab 7: Structs and Methods**
    a. Define a struct for a simple model (e.g., a Car or Book)
    b. Create methods for struct types
2. **Lab 8: Implementing Interfaces**
    a. Create and use interfaces in a program
    b. Use type assertions to handle empty interfaces

Day 5: Concurrency in Go

- Objective: Learn Go's powerful concurrency model using goroutines and channels.

Morning Session (Theory)

1. **Goroutines**
    a. Introduction to concurrency in Go
    b. Creating and using goroutines
    c. Goroutine life cycle
2. **Channels**
    a. Working with channels for communication between goroutines
    b. Buffered vs. unbuffered channels
    c. Using the select statement for multiplexing

Afternoon Session (Lab)

1. **Lab 9: Goroutines in Action**
    a. Create concurrent functions using goroutines
    b. Simulate concurrent tasks (e.g., downloading files in parallel)
2. **Lab 10: Using Channels**
    a. Build a program where multiple goroutines communicate via channels
    b. Use select to manage multiple channel inputs