Go Language Overview for Non-Programmers

Course Summary

Description

This is a high level survey course for those who need an understanding of the Go programming language and concepts, but are not planning to do any Go programming themselves or who are novice programmers. This audience would include developers in other languages who need a high level understanding of Go, testers who will be working with Go programmers, documentation specialists, team leads, integration engineers, and anyone evaluating the language. This course addresses the reality that not everyone who needs to understand Go is going to be a Go developer.

The lectures focus on the concepts used in Go programming, with easily understood examples illustrating the approaches and techniques that exemplify the "Golang" way of both solving programming problems and writing code. The hands on labs are intended to clarify and reinforce the concepts presented in the lectures, and are designed so that they can be accomplished by everyone including those with minimal coding skills.

Because Go is more than just a language but is an integrated development and software build system designed for modern large scale and continuous development projects, the course also explores the Go ecosystem including the Go command, build tools, utilities and standard libraries. These are all presented in the context of how the language developers intended them to be used, demonstrated with typical use cases.

Side by side comparisons of Go with other important languages (Java, C++, C, Python, etc) provide an evaluation of the relative strengths and weaknesses of Go and these other languages.

The purpose of the course is not to teach students to become Go programmers but to provide a high level overview of Go language structure, concepts, practices and the Go environment. The overall objective of the course is for students to be able to read a Go program listing and understand what the code is doing.

Objectives

After taking this course, students will be able to understand:

- The problems in modern software development that led to the creation of Go, how the design of Go addresses those problems, as well the design philosophy of Go.
- How the standard Go development environment is set up and why it is done this way.
- The basic Go utility commands (go build, go format, go test, etc).
- What happens in a Go build environment as code moves from source to executable.
- The basic structure and architecture of a Go application.
- How Go data types and variables work, including how they are similar and different that other programming languages.
- How Go functions, data structures and how error handing works.
- How the Go control structures are different from other programming languages.
- How Go implements object oriented programming and functional programming.
- How Go implements concurrency.
- The basic Go libraries and what they are used for.
- How Go incorporates external libraries.
- How Go applications are designed and their architecture.
- How Go compares to other programming languages

Go Language Overview for Non-Programmers

Course Summary (cont'd)

Topics

- The software development crisis – legacy tools in a modern world.
- The basic goals of Go – an efficient software development environment and language designed for a DevOps, Big Data, high performance hardware type of environment.
- The Go build space – standardized and scalable.
- The Go tools – what happens when using build, test, format, run, install, get and other options.
- The structure of a Go application and the Go programming model.
- Packages, imports and remote imports.
- Go data types and how they are different from other languages.
- How variables and assignments work in Go, variable scopes.
- How Go implements basic control structures (loops, conditionals, etc).
- The Go function model – multiple returns, returning error objects.
- Go data structures – maps, arrays, slices, etc.
- How Go manages strings – UTF-8 and runes.
- Error handling in Go and deferred functions.
- Structs, methods and interfaces in Go.
- Implementing object oriented programming.
- Functional programming in Go – function as first class objects, anonymous functions.
- Concurrency as an integrated part of Go – goroutines and channels.
- The Go way of approaching application design – Golang patterns.
- Basic Go libraries – networking, I/O, internet and integration with legacy C code.
- Most popular Go add on libraries.
- Feature and usage comparison of Go with Java, C, C++, Python, Rust and others.

Audience

This course is designed for those who need to understand how Go works, how to read Go code or otherwise get a fundamental understanding of Go but who do not need to master learning how to write Go code. The course is appropriate for testers, integrators, documentation specialists, team leads and others who will be working closely with Go developers or their Go code. The course is also appropriate for developers who are evaluating Go compared to other languages or for novice programmers.

Prerequisites

Before taking this course, students should have a basic understanding of programming concepts. Hands-on programming experience in a programming language is helpful but not essential.

Duration

Two days