Loc Nguyen

Professor Raheja

CS.4080.03

Evaluation of Go Programming Language

Go or Golang is a newly established language started development in September 2007 with three original developers: Swiss computer scientist Robert Griesemer, who worked on Google V8 JavaScript engine, Sawzall language, and Java HotSpot virtual machine; Canadian programmer Rob Pike, who worked at Bell Labs on the Unix team and was involved with the creation of Plan 9, Inferno OS, Limbo programming language, and co-creator for UTF-8; and lastly, American computer scientist Ken Thompson, who also worked at Bells Labs and was involved in B and C programming language, Plan 9, and was dubbed “Father of Unix” alongside Dennis Ritchie for his work on designing and implementing the original Unix OS.

The creators of Go wanted something that was expressive like Python or JavaScript but lightweight and efficient like C or C++. The triangle of tribulation of choosing only two out of three conditions of efficient compilation, efficient execution, or ease of programming, meant that developers had to sacrifice and compromise. For example, they would choose ease of programming over safety or efficiency by using dynamically typed language like Python. The moment was ripe for Go to address these flaws by combining the ease of dynamically type language with safety and efficiency of statically typed, compiled language.

To maintain efficiency, Go is excellent at compilation speed, boasting a compilation speed that’s even faster than C and a lot of compiled language. This allows programmers and developers to quickly iterate, test, and deploy code to production which decreases downtime versus something like Java which takes a while to compile huge programs. To achieve this, creators of Go decided to make Go an imperative and procedural programming language, mixing in a little bit of functional programming language features such as first class and lambda functions. That means there are no classes, objects, or features like inheritance from object-oriented programming languages. However, you can mimic some functionalities with interface and struct in Go.

First, like C, Go must import packages like “fmt” which handles input/output stream operation or other relevant packages such as “math” for specialized mathematical operations. The main function must be defined but with minimal boilerplate code, like Python, which increases writability. Syntax and scopes in Go is eerily similar to what we see in Python and C/C++ where it supports single-line multivariable declaration or initialization of the same type, out of scope once function finishes, and type inferencing where, instead of the keyword “var” follow by “=” to specify the variable type, Go uses “:=” for assignment operation and guess the type based on assigned value. It’s important to note that variables are still statically typed and checked at compiled time, unlike Python.

Data types in Go include basic (numbers, strings, Boolean), aggregate (array and structure), reference (slice, map, function, pointers, and channel), and interface. The three basic data types are typical and almost universal among programming languages. Aggregate type struct store collection of heterogeneous fields into a single field and array is mutable, fixed size, and less popular than slices. Function in Go is very useful and intuitive, allowing for “naked” return statement where return variables as specified before function is defined. Same for pointers, which may be used to access field values in a struct, mimicking OOP properties. Slices are dynamically sized and are more popular than arrays due to its flexibility so operations on slices are done to speed up execution and decrease memory usage. Map behaves like a dictionary in Python where it maps a key to a value, but a notable keyword “make” must be explicitly called to initialize a map “object” or a channel. The make keyword is implicit called for pointers, functions, slices, arrays, and various other structures but in this case, it must be called explicitly. This decreases uniformity and regularity in Go because of special cases such as this.

The most special feature of Go is channel, a typed conduit which you can send and receive values using the channel operator <-. Channels are often used in goroutines, a way to speed up program execution speed. Instead of using