Matt Burton

CSC-4510

12/8/23

**Google’s Go Language**

**Executive Summary**

Google's Go language, commonly referred to as Golang, was created to meet the need for a more efficient, scalable, and readable language. Officially released in 2012, Go was designed to improve software development, offering simplicity, concurrent programming support, and scalability. Its simple and powerful features, including goroutines and fast compilation, make it an enticing language for large-scale system development. This paper introduces the structure, features, strengths, weaknesses, and practicality of Go. It highlights Go’s straightforward syntax, concurrency support, robust standard library, and its unique approach to modules, testing, and packages.

**Introduction**

This essay aims to explore the core components of Go, elaborating on its design philosophy, features, and practical aspects. It encapsulates not only the language's robustness but also my firsthand experiences navigating its functionalities and addressing challenges faced during its utilization.

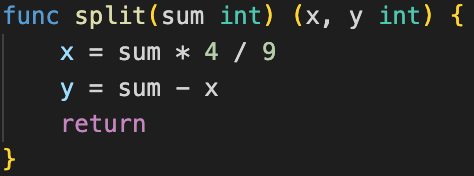
**Tutorial**

My experience with Go started with the many tutorials offered on [go.dev](https://go.dev/) so I could learn its fundamental structure and syntax. Go emphasizes simplicity and readability so picking up the syntax was relatively straightforward. Newer concepts such as modules, packages, and function values proved more difficult to pick up, but the provided documentation helped to overcome the initial confusion. When using Go, it is important to understand that every program belongs to a package defined at the top of the file.



This package is then accessible by other packages by importing it as a module. By default, the main package will run first, and make calls to its imported packages for further execution.

Much of Go’s basic features are similar to familiar languages like C, C++ and Java. Go natively supports functions, arguments, and return values:



variables, constants, and type conversion:

looping, if and switch statements, structs, pointers, arrays, generics, maps and more. However it doesnt stop there, Go differs from languages like C in many ways and offers concepts that I’ve never seen before. Most notably, Go implements slices for dynamic arrays, methods in place of classes, and goroutines to handle threading.

**Analysis**

**Compiler Challenges**

**Conclusion**