About Go

Go is a statically typed, compiled programming language designed at Google by Robert Griesemer, Rob Pike, and Ken Thompson. Go is syntactically similar to C, but with memory safety, garbage collection, structural typing, and CSP-style concurrency.

Go is an open-source programming language focused on simplicity, reliability, and efficiency.

Go was originally built for programs related to networking and infrastructure. It was intended to replace popular high-performance server-side languages like Java and C++. Today, Go is used for a variety of applications:

Go is popular for cloud-based or server-side applications.
DevOps and site reliability automation are also popular ways to use Go.
Many command-line tools are written in Go.
Go is used in the world of artificial intelligence and data science.
Some use Go from micro-controller programming, robotics, and games.
However, Go really shines the most when it comes to infrastructure. Some of the most popular infrastructure tools today are written in Go — such as Kubernetes, Docker, and Prometheus.

Go is great when it comes to performance. It was designed for automation at a large scale so Go makes it relatively easy to write high-performing applications.

According to the 2020 Stack Overflow Developer Survey, Go is one of the most loved languages by developers who use it. Go's own 2020 developer survey seems to agree, with a 92% satisfaction rating by Go users.

Go Vs Python

Python and Go are different, generally serving different purposes. Python is the primary language among data scientists, where Go is the language for server-side commands. Go is the language to use to run software. It is the faster language, performing at Java and C++ speeds.

Python is the language to use for readable, shareable code—hence the large community around it.

Technically, Go is a procedural, functional language built for speed, and Python is an object-oriented, imperative, functional, and procedural language. Go supports concurrency, the ability of an algorithm to run its steps out of order, and Python doesn't.

In short, Python is good when you're working with data and the audience is people, wherein, Go is while working with servers.

Python and Go both have simple syntax and first-party support from all major cloud providers.

Both Go and Python are easy to get started with for beginners and are relatively easy to learn. There's debate over which is easier. Go is a simpler language and may be mastered more quickly, but some find getting started more difficult than Python, which takes longer to master as there's more to learn.

Go is a new kid on the block compared to Python, and it was designed to be fast. Go is faster than Python. Much faster.

Python tends to dominate in data science; Go is perfect for system programming.

As the senior language, Python has a more extensive library and community built up around it. Python's dynamic typing can make it better than Go for quick prototyping.
It can be easier to run applications at scale with Go. Go was built by Google to solve problems at a Google-sized scale making it ideal for working on large concurrent applications. Go supports concurrency, or the ability to run more than one program/task simultaneously. Python doesn't.

Go Vs Java

Java is the older and more widely used programming language. It is object-oriented, has a larger community—thus library, and relies on the Java virtual machine (JVM). Wherein, Go, or Golang, is newer, supports concurrency, is more readable, and is not object-oriented.

Go is faster than Java on almost every benchmark. This is due to how it is compiled: Go doesn't rely on a virtual machine to compile its code. It gets compiled directly into a binary file. ... Because Go does not have the VM, it is faster.

Go and Java are both C-family languages which means they share a similar language syntax. That's why Java developers often find reading Go code fairly easy and vice versa. Go does not use a semicolon (';') symbol at the end of a statement though except in occasional cases where it is needed. To me, the line-delimited statements of Go feel clearer and more readable.

Go and Java also both use one of my favorite features, garbage collector (GC), to help prevent memory leaks. Unlike C++, C-family programmers have to worry about memory leaks and garbage collector is one of those features that automates memory management and therefore simplifies their job.

Golang is not an OOP language. At its core, Go lacks the inheritance of Java because it does not implement traditional polymorphism by inheritance. In fact, it has no objects, only structures. It can simulate some object-oriented patterns by providing interfaces and implementing the interfaces for structures. Also, you can embed structures to each other but embedded structures

do not have any access to the hosting structure's data and methods. Go uses composition instead of inheritance in order to combine some desired behavior and data.

Go is an imperative language and Java tends to be a declarative language. In Go, we don't have anything like dependency injection; instead, we have to wrap up everything together explicitly. That's why the recommended approach to programming in Go is to use as little magic as possible. Everything should be really obvious for an external code reviewer and a programmer should understand all the mechanisms for how the Go code uses the memory, file system, and other resources.

On a benchmark test to calculate factorials, by Sunny Radadiya, Go performed better than Java.

Go speed:

Output:

Factorial	Time To calculate factorial
10000	0.03 seconds
50000	0.41 seconds
100000	2.252 seconds
500000	68.961 seconds
1000000	224.135 seconds

Java speed:

Output:

Factorial	Time To calculate factorial
10000	0.112 seconds
50000	1.185 seconds
100000	2.252 seconds
500000	89.500 seconds
1000000	385.868 seconds