How well does this language translate into other languages, such as Java, Python, Ruby, etc.?

Go is not as full-featured as some other languages, so I wouldn’t use it as a starter language if you’re interested in CS theory. If you know what you’re doing, you can move fast like in Python. If you have experience with Java, you should be fine writing Go although there are some significant differences between the two; experience with a GC would be beneficial, however. Translating codebases to/from Go would be difficult, and the effort should be evaluated on a case-by-case basis. That being said, many companies have translated their codebases to Go.

Why is Go's method of error handling better?

The creators of Go believed exception handling, i.e. try-catch, created convoluted code. For that reason, they treat errors as a possible return value, as most of the time an error isn’t a truly “exceptional” condition, i.e. failing to open a file. By treating errors as values, Go is inherently more resilient and less likely to fail catastrophically. Go does have panic and recover methods for truly exceptional events, but this error handling scheme forces the user to go out of the way to throw these program-breaking exceptions, rather than making that the default common case, like with C exceptions.

A little more history would have been nice to hear. What kind of problems was Go looking to solve?

When Go was created, C++ and Java dominated production software and GitHub didn’t exist. Seeing the rise in universal multiprocessors, the creators wanted to create a simpler, highly-parallelizable programming language ideal for large codebases. They wanted to remove the barriers programmers faced with traditional languages by enabling tooling and automation of mundane tasks, and garbage collection.

I know go wasn’t recommended to be used everywhere, so where can it not be used?

I wouldn’t recommend Go to be used in any codebase that’s heavily prone to errors and relies a lot on polymorphism. One of Go’s major weaknesses is interfaces and how difficult it makes polymorphism to function as we’re used to. Furthermore, error-checking in Go can get confusing and incredibly verbose. Especially if you’re not taking advantage of multiple threads, these cons may outweigh the positives of writing a program in Go, as opposed to some other high-level language.

What are the limitations of Go compared to older languages like C++?

No pointer arithmetic -- you can’t do as many unsafe procedures as you can in lower-level languages. Furthermore, it’s difficult to manage your own memory due to the garbage collector. Perhaps the biggest and most jarring limitation is the lack of generics. It would be difficult to translate generics from a Java or C++ codebase to function properly in Go.

What is Go commonly used for (like what does Google and Docker use it for), i.e. games?

While Go is a general purpose language, it’s most commonly used for developing scalable web services. It replaced Java and C++ in Google’s own software stack. So it is commonly used for any large-scale, distributed systems, for example: streaming on Netflix and Twitch, Google, Uber maps, chat servers, and news sites. The common thread here being each of these services need to be scalable and adapt to high levels of traffic.

Why don't we learn this language at Binghamton??

Good question.

Does Go use any sort of deadlocks or something similar when u access shared resources?

When a goroutine accesses a value from a channel it automatically blocks that channel from being accessed by another goroutine. So yes, channels are using locks under the hood.