Go Variables

Ian Harris

December 14, 2018

1 Go Variables

So now we're going to start talking a little bit about the Go language, will broached the topic of variables and talk a little bit about those variables that are in every high level language and we'll just see how Go implements it. A lot of this is very similar to what you've seen in other languages, some things are a little bit different.

1.1 Variable Names

So first there's naming. Every- you need names. Names are variables for functions, you need names to refer to things in your code. So names for variables and things like that, they need to start with a letter. They can have any number of letters and digits and underscores, they are case sensitive in Go and you can't use keywords. There's a list of keywords you can google these or look them up but "if", "case", "package" all the different keywords, the language you can't use those as the names.

1.2 Variables Reference Data with Name and Type

So, variables are basically data stored in memory somewhere. And every variable has to have a name and a type.

1.2.1 Variables Declare Name and Type

So all variables have to have a declaration, specifies the name, and the type of the variable. So, here's a really simple variable declaration. Just as `var x int`. So `var` is the keyword for a **declaration** of a variable. After that I have the name. So my name variable is called `x` and then after that I have the type: `var x int`. That's it. That's a declaration of this variable `x` since it's an integer. And the compiler needs to know what type of variable it is, what the type is, so it knows how much space to allocate, what operations to perform that type of thing. You can declare many on the same line if you want to, just comma-separated. So `var x, y int` and you can do that as much as you want.

1.2.2 Types

So variables have types. *Type defines the values that a variable can take and the operations that can be performed on that variable.* So for instance common types, basic types, integer, floating point, strings.

Integer Type Integers, the data, the values that they can take are only integral values right? They are integers and the operations you can perform are integer arithmetic, plus, minus, times that sort of thing and there are a set of other ones we'll talk about them a little more detail.

Float Type Floating point, those are the data that they can have, the values they can have or fractional like decimal values and there you have a set of operations, arithmetic operations. Actually they look superficially the same as the integer operations, plus, times, divide but they may actually be implemented with different hardware right? Because floating point division say is significantly more complicated than integer division. So there's oftentimes as special hardware just for floating point divide things like this. We don't have to know that as programmers but the machine has to know which operation to map it to.

String Type Then strings. So strings they're a sequence of bytes represented in unicode and we'll get into that later. But it's a sequence of bytes, that's the type of data the values it can take on and then the operations you can perform on strings. There are many of them. String comparison, string search, concatenation, all sorts operations that you can you can perform on a string.

1.2.3 What Types Tell

But the point is the type specifies these things. It specifies what data the variable can hold and how big that data can be right? Because you need to know how much space in memory you're going to need to allocate for this. The compiler needs to know that and also what operations are going to be performed on it. So what that's for is eventually the compiler is going to have to take these these operations that you type and go and compile them into machine code instructions for whatever the hardware platform is, and those machine code instructions can be different depending on the type. So for instance you can easily have an add for an integer, an integer add up machine code instruction which is different than a floating point add right? Integer division which is different than floating point division and so on. So this is why the compiler needs to know the type so it knows how to do the compilation, how to convert it into machine code.