Generics

Go supports generic programming using type parameters. This lesson shows some examples for employing generics in your code.

The Go Authors

https://golang.org

*** Type parameters**

Go functions can be written to work on multiple types using type parameters. The type parameters of a function appear between brackets, before the function's arguments.

```
func Index[T comparable](s []T, x T) int
```

This declaration means that `s` is a slice of any type `T` that fulfills the built-in constraint `comparable`. `x` is also a value of the same type.

`comparable` is a useful constraint that makes it possible to use the `==` and `!=` operators on values of the type. In this example, we use it to compare a value to all slice elements until a match is found. This `Index` function works for any type that supports comparison.

*** Generic types**

In addition to generic functions, Go also supports generic types. A type can be parameterized with a type parameter, which could be useful for implementing generic data structures.

This example demonstrates a simple type declaration for a singly-linked list holding any type of value.

As an exercise, add some functionality to this list implementation.

*** Congratulations!**

You finished this lesson!

You can go back to the list of [[/tour/list][modules]] to find what to learn next, or continue with the [[javascript:click('.next-page')][next lesson]].