Part 12: Variadic Functions

25 APRIL 2017

Welcome to the part 12 of Golang tutorial series.

What is a variadic function?

A variadic function is a function that can accept variable number of arguments.

Syntax

If the last parameter of a function is denoted by `...T`, then the function can accept any number of arguments of type `T` for the last parameter.

Please note that only the last parameter of a function is allowed to be variadic.

Examples

Have you ever wondered how the append function which is used to append values to a slice accepts any number of arguments. Its because its a variadic function. `func append(slice []Type, elems ...Type) []Type` is the definition of append and `elems` is a variadic parameter.

Lets create our own variadic function. We will write a simple program to find whether a integer exists in a input list of integers.

```
1  package main
2
3  import (
4      "fmt"
5  )
6
7  func find(num int, nums ...int) {
8      fmt.Printf("type of nums is %T\n", nums)
9      found := false
10     for i, v := range nums {
11         if v == num {
12             fmt.Println(num, "found at index", i, "
13             found = true
14         }
15     }
16     if !found {
17         fmt.Println(num, "not found in ", nums)
18     }
19     fmt.Printf("\n")
20 }
21 func main() {
22     find(89, 89, 90, 95)
23     find(45, 56, 67, 45, 90, 109)
24     find(78, 38, 56, 98)
25     find(87)
26 }
```

In the above program, `func find(num int, nums ...int)` accepts variable number of arguments for the `nums` parameter. Internally `...int` is considered as a slice. The type of `nums` in this function is `[]int` (integer slice).

In line no. 10, the `for` loop ranges over the `nums` slice and prints the position of `num` if it is present in the slice. If not it prints that the number is not found.

The above program outputs,

```
type of nums is []int
89 found at index 0 in [89 90 95]

type of nums is []int
45 found at index 2 in [56 67 45 90 109]

type of nums is []int
78 not found in  [38 56 98]

type of nums is []int
87 not found in  []
```

In line no. 25 of the above program, the find function call has only one argument. We have not passed any argument to the variadic `nums ...int` parameter. This is perfectly legal and in this case `nums` is a `nil` slice with length and capacity 0.

Passing a slice to a variadic function

We have already mentioned that `...T` is internally a slice of type `T`. If that is the case, will passing a slice to a variadic function work? Lets find out from the next example.

```
1  package main
2
3  import (
4      "fmt"
5  )
6
7  func find(num int, nums ...int) {
8      fmt.Printf("type of nums is %T\n", nums)
9      found := false
10     for i, v := range nums {
11         if v == num {
12             fmt.Println(num, "found at index", i, "
13             found = true
14         }
15     }
16     if !found {
17         fmt.Println(num, "not found in ", nums)
18     }
19     fmt.Printf("\n")
20 }
21 func main() {
22     nums := []int{89, 90, 95}
23     find(89, nums)
24 }
```

In line no. 23, instead of passing variable number of arguments to the last parameter of the find function, we are passing a slice. This is not allowed in Go. You cannot pass a slice to a function that is expecting a variadic parameter. The above program will throw error `main.go:23: cannot use nums (type []int) as type int in argument to find`

There is a syntactic sugar which can be used to pass a slice to a variadic function. You have to suffix the slice with `...` If that is done, the slice is expanded to its individual elements and passed to the function and the program will work.

In the above program if you replace `find(89, nums)` in line no. 23 with `find(89, nums...)`, the program will compile and output

```
type of nums is []int
89 found at index 0 in [89 90 95]
```

Thats it for variadic function. Thanks for reading. Please leave your valuable feedback and comments. Have a good day.

Next tutorial - Maps

Naveen Ramanathan

iOS developer at dietco.de and Golang enthusiast.