A Slice of Maps

This lesson discusses how to make slices of maps.

WE'LL COVER THE FOLLOWING

Explanation

Suppose we want to make a slice of maps. We must use `make()` *two* times, first for the slice, then for each of the map-elements of the slice. To access a specific key-value pair from a map, you have to use an iterator to specify which map from the slice of maps is required.

For example, if we have a slice of maps called `maps`, and we want to set a value `v` with key `1` from map `i`, we'll do something as follows:

```
maps[i][1] = v
```

Simply writing `maps[1] = v` won't work, because it will initialize map-variables, and will be lost on the next iteration.

Look at the following implementation of how to make a slice of maps in Go.

```
package main
import (
"fmt"
)

func main() {

  // Version A:
  items := make([]map[int]int, 5)
  for i := range items {
    items[i] = make(map[int]int, 1)
    items[i][1] = 2 // This 'item' data will not be lost on the next iteration
  }
  fmt.Printf("Version A: Value of items: %v\n", items)
  // Version B: NOT GOOD!

  items2 := make([]map[int]int, 5)
  for _, item := range items2 {
    item = make(map[int]int, 1) // item is only a copy of the slice element.
    item[1] = 2 // This 'item' will be lost on the next iteration.
  }
  fmt.Printf("Version B: Value of items: %v\n", items2)
}
```

Slice of Maps

In the code above, in `main` at line **9**, we make a slice of maps: `items := make([]map[int]int, 5)`. The declaration of `items` shows that it contains **5** maps. For each map, its keys will be of *int* type and the values associated with its keys will also be of *int* type. Now, there is a for loop at **line 10**, which iterates through all **5** places and places a map with the key `1` (at **line 11**). For every map, the value **2** is assigned with the key `1` (at **line 12**) as: `item[i][1] = 2`.

Now let's try a separate version. In the code above, in `main` at line **19**, we make a slice of maps as: `items2 := make([]map[int]int, 5)`. The declaration of `items2` shows that it contains **5** maps. For each map, its keys will be of *int* type and values associated with its keys will also be of *int* type. Now, there is a for loop at **line 20**, which iterates through all **5** places and places a map with the key `1` (at **line 21**). For every map, the value **2** is assigned with the key `1` (at **line 22**) as : `item[1] = 2`. The value will be lost in the next iteration because `item` is the copy of the slice; proper indexing is required as done in Version 1 at **line 12**.

That's how you can make slices of the type *map* and access them when needed. The next lesson covers some more concepts on modifying the maps.