MAVENIR™
Trust the Future

# Golang Training Pointers

24/01/2023

Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Maps

## Overview

- Map is an unordered collection of key/value pairs
  - Known as associative arrays in Php, hash tables in Java, or dictionaries in Python
- Used to look up a value by its associated key
- Store values into the map based on a key
- Map retrieves data quickly based on the key
- Key works like an index, pointing to the value
- A map is implemented using a hash table
- Maps are not safe for concurrent use

MAVENIR

Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Overview

- Maps are unordered collection
- Can not predict the order in which the key/value pairs will be returned
- Every iteration over a map could return a different order.


Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Initialization: Empty map

- Maps are written with curly brackets, and they have keys and value

Syntax:

var <map-variable-name> = map[key-type]value-type{}

e.g. var employee = map[string]int{}

employee: Map variable name

string : Data type of key

int: data type of value


Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Declaration with make

- make function takes as argument the type of the map and it returns an initialized map

Syntax

var <map-variable> = make (map[key-type]value-type)

var serviceFQDNs = make(map[string][string])

serviceFQDNs["auth-url] = "https://auth.mavenir.com:443/authenticate"

serviceFQDNs["db-conn"] = "https://auth.mavenir.com:5656/database"


Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## length

> Determine how many items (key-value pairs) a map has

  – Use built-in len() function

E.g

```
var employee = make(map[string]int)

employee["Vijay"] = 10

employee["Anil"] = 20

employeeList := make(map[string]int)

fmt.Println(len(employee))      // 2

fmt.Println(len(employeeList))  // 0
```

MAVENIR

Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Accessing Items

> Can access the items of a map, by referring to its key name, inside square brackets

Eg.

```
var employee = map[string]int{"Vijay": 10, "Anil": 20}

fmt.Println(employee["Mark"])
```


Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Adding Items

- Item can be added by using a new index key and assigning a value to it

E.g.

```
var employee = map[string]int{"Anil": 10, "Vijay": 20}

fmt.Println(employee) // Initial Map


employee["Vishal"] = 30 // Add element

employee["Sanjay"] = 40
```


Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Update values

> Update the value of a specific item by referring to its key name

E.g.

```
var employee = map[string]int{"Vijay": 10, "Vinay": 20}

fmt.Println(employee) // Initial Map


employee["Vijay"] = 50 // Edit item or rewrite item
```


Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Delete

> Built-in delete function deletes an item from a given map associated with the provided key.

E.g.

```
var employee = make(map[string]int)
    employee["Vijay"] = 10
    employee["Vishal"] = 20

    fmt.Println(employee)

    delete(employee, "Vijay") // deletes key-value pair "Vijay" 10
    fmt.Println(employee)
```

MAVENIR

Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Iterate over a Map

- The for…range loop statement can be used to fetch the index and element of a map.

```
var employee = map[string]int{"Vijay": 10, "Mayur": 20,
    "Vishal": 30, "Arun": 40, "Ajay": 50}
for key, element := range employee {
    fmt.Println("Key:", key, "=>", "Element:", element)
}
```

- Maps can't be iterated using len loop


Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Truncate map

- There are two ways to clear all items from a Map.
  - Deleting Items by iterating
  - Reinitializing map using make()

```
var employee = map[string]int{"Vishal": 10, "Ajay": 20,
    "Prabhat": 30, "Kailash": 40, "Manasa": 50}


// Method - I
for k := range employee {
   delete(employee, k)
}
employee = make(map[string]int)
```

MAVENIR

Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Sort Map Values

- Sort the key values of a map, you need to store them in Slice and then sort the slice

```
unSortedMap := map[string]int{"India": 20, "Canada": 70, "Germany": 15}
 // Int slice to store values of map.
values := make([]int, 0, len(unSortedMap))
for _, v := range unSortedMap {
  values = append(values, v)
 }
// Sort slice values.
sort.Ints(values)
// Print values of sorted Slice.
for _, v := range values {
  fmt.Println(v)
}
```

MAVENIR

Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Sort Map

- Sort Map By keys
- Sort Map By Values

MAVENIR

Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Merge Map

```
first := map[string]int{"a": 1, "b": 2, "c": 3}

second := map[string]int{"a": 1, "e": 5, "c": 3, "d": 4}


for k, v := range second {

        first[k] = v

}


fmt.Println(first)
```


Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Allowed Value types

- Map key can be any type that is comparable
  - boolean
  - numeric
  - string,
  - pointer
  - channel
  - interface types
  - structs – if all it's field type is comparable
  - array – if the type of value of array element is comparable
- Slice, Map, Function are not allowed key types


Copyright Mavenir 2022. Proprietary and Confidential.

# Golang: Map

## Operations

- Add a key-value pair
- Update a key
- Get the value corresponding to a key
- Delete a key-value pair
- Check if a key exists


Copyright Mavenir 2022. Proprietary and Confidential.