Day 24 - Go (Golang) Basics - Maps

What Is a Map?

A map is a data structure that stores **unordered key-value pairs**. It's like a dictionary where you use a key to look up a value.

- Maps are used to retrieve values using a key.
- Go maps are implemented using **hash tables**.
- Very efficient for adding, accessing, and deleting data.

Declaring a Map

Syntax:

`var <mapName> map[<keyType>]<valueType>`

Example:

```go
package main

import "fmt"

func main() {
    var fruitColors map[string]string
    fmt.Println(fruitColors) // nil
}
```

This creates a **nil map**. You cannot add items to a nil map, it will panic.

Common Error (nil map)

```go
package main

import "fmt"

func main() {
    var fruitColors map[string]string
    fruitColors["apple"] = "red"  // This will panic
    fmt.Println(fruitColors)
}
```

Initializing a Map

Using a map literal:

```go
package main

import "fmt"

func main() {
    fruitColors := map[string]string{
        "apple":  "red",
        "banana": "yellow",
        "grape":  "purple",
    }
    fmt.Println(fruitColors)
}
```

Using `make()`:

```
package main

import "fmt"

func main() {
    fruitColors := make(map[string]string)
    fruitColors["orange"] = "orange"
    fmt.Println(fruitColors)
}
```

With capacity (optional):

```
fruitColors := make(map[string]string, 10)
```

Accessing Map Elements

```
fmt.Println(fruitColors["apple"])  // red
```

Safe access with a check:

```
if color, found := fruitColors["apple"]; found {
    fmt.Println("Color:", color)
} else {
    fmt.Println("Fruit not found")
}
```

Adding or Updating a Value

```
fruitColors["kiwi"] = "green"
```

Deleting a Key

```
delete(fruitColors, "banana")
```

Getting Length of Map

```
fmt.Println(len(fruitColors)) // prints number of key-value pairs
```

Looping Over a Map

```
for fruit, color := range fruitColors {
    fmt.Println(fruit, "=>", color)
}
```

Truncating a Map

Option 1: Delete keys manually

```
for k := range fruitColors {
```

```go
    delete(fruitColors, k)
}
```

Option 2: Reinitialize

```go
fruitColors = map[string]string{}
```

Copying Maps

Maps are **reference types**. Assigning one map to another allows them to share the same data.

Example:

```go
a := map[string]string{"apple": "red"}
b := a
b["apple"] = "green"
fmt.Println(a["apple"]) // Output: green
```

To truly copy:

```go
a := map[string]string{"apple": "red"}
b := make(map[string]string)
for k, v := range a {
    b[k] = v
}
b["apple"] = "green"
fmt.Println(a["apple"]) // Output: red
fmt.Println(b["apple"]) // Output: green
```