Go Array

In Go, an array is a homogeneous data structure (Fix type) and has a fixed-length. The type can be anything like integers, string or self-defined type.

- ➔ The items in the array can be accessed through index.
- ➔ It starts with zero
- ➔ The number of items in the array is called the length or size of an array.
- ➔ It is fixed and must be declared in the declaration of an array variable.

Syntax:-

```
   var identifier [length] type
```

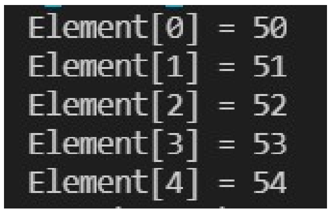
Example:-

```
    var arr_name [5] int
```

One-Dimensional Array Example :-

```go
package main

import "fmt"

func main() {
    var arr [5]int
    var i, j int

    for i = 0; i < 5; i++ {
        arr[i] = i + 50
    }

    for j = 0; j < 5; j++ {
        fmt.Printf("Element[%d] = %d\n", j, arr[j])
    }
}
```

Output:-



```
Element[0] = 50
Element[1] = 51
Element[2] = 52
Element[3] = 53
Element[4] = 54
```

Multi-Dimensional Array

Multi dimensional arrays is simply a list of one-dimensional arrays. Or we can say it is Array of Array.

Syntax :-

```
var arrayName[x][y] variable_type
```

Example :-

```
arr = [3][4] int
```

Initializing Two-Dimensional Arrays

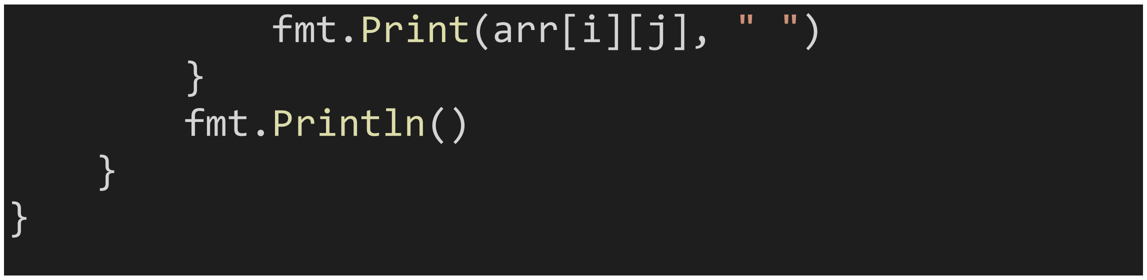
```
arr = [2][3] int{
    {2,4,6}, // initialization of row of index
zero.
    {8,10,12} // initialization if row of index 1
}
```

Accessing Two-Dimensional Arrays

```
int val = arr[1][2] // accessing second element of
1st row of array
```

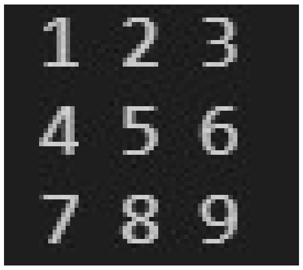
Multi-Dimensional Array Example :-

```
package main

import "fmt"

func main() {
    // an array with 3 rows and 3 column
    var arr = [3][3]int{
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9},
    }

    var i, j int
    // Output of each array elements
    for i = 0; i < 3; i++ {
        for j = 0; j < 3; j++ {
```



```
            fmt.Print(arr[i][j], " ")
        }
        fmt.Println()
    }
}
```

Output:-



```
1 2 3
4 5 6
7 8 9
```