Working with arrays

This lesson explains arrays, multidimensional arrays and how to print them in GO

WE'LL COVER THE FOLLOWING ^
Arrays
Printing arrays
Multi-dimensional arrays

Arrays

The type [n]T is an array of n values of type T.

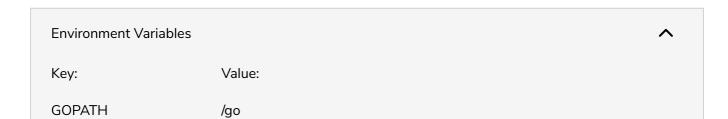
The expression:



declares a variable a as an array of ten integers.

An array's length is part of its type, so arrays *cannot* be resized. This seems limiting, but don't worry; Go provides a convenient way of working with arrays.

Let's take a look at an example below:

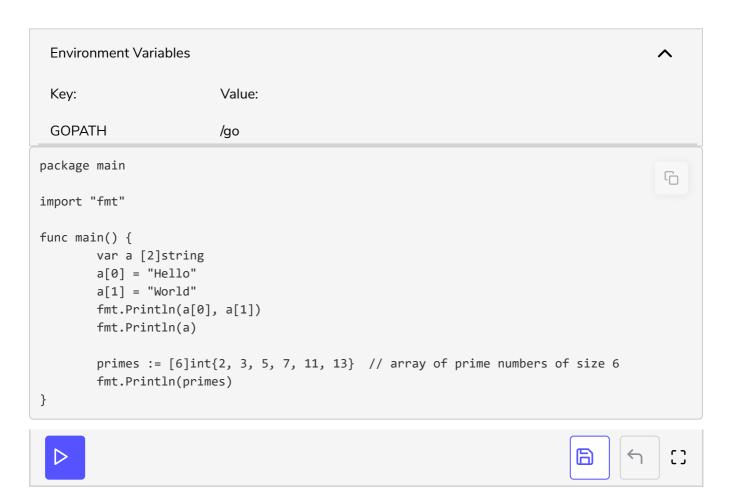


```
package main

import "fmt"

func main() {
   var a [2]string    //array of size 2
   a[0] = "Hello"    //Zero index of "a" has "Hello"
   a[1] = "World"    //1st index of "a" has "World"
   fmt.Println(a[0], a[1]) // will print Hello World
   fmt.Println(a)    // will print [Hello World]
}
```

Below is another example to further help understand the concept:



You can also set the array entries as you declare the array:

Environment Variables		^
Key:	Value:	
GOPATH	/go	
package main		(f)
import "fmt"		

```
func main() {
    a := [2]string{"hello", "world!"}
    fmt.Printf("%q", a)
}
```

Finally, you can use an **ellipsis** to use an implicit length when you pass the values:



Printing arrays

Note how we used the fmt package using Printf and used the %q "verb" to print each element quoted.

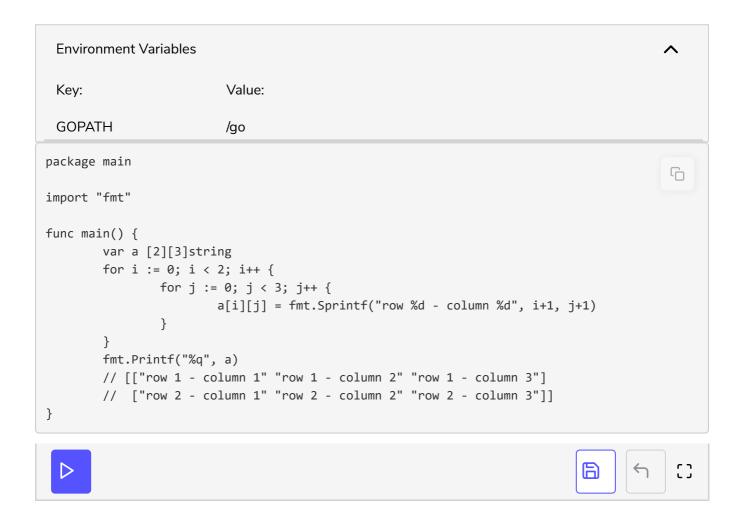
If we had used **Println** or the **%s** verb, we would have had a different result:



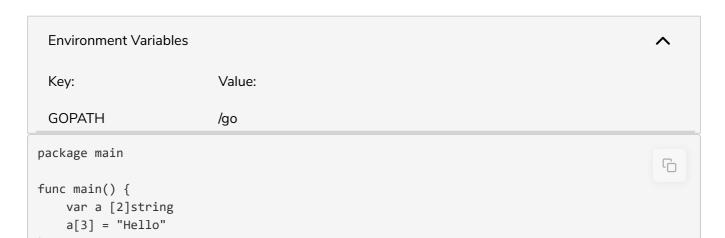
```
fmt.Printf("%s\n", a)
    // [hello world!]
    fmt.Printf("%q\n", a)
    // ["hello" "world!"]
}
```

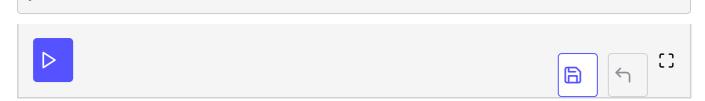
Multi-dimensional arrays

You can also create multi-dimensional arrays:



Trying to access or set a value at an index that doesn't exist will prevent your program from compiling, for instance, try to compile the following code:





Compiler will generate Error

That's because our array is of length 2 meaning that the only 2 available indexes are **0** and **1**. Trying to access index **3** results in an error that tells us that we are trying to access an index that is *out of bounds* since our array only contains 2 elements and we are trying to access the 4th element of the array.

Slices, the type that we are going to see in the next lesson is more often used, due to the fact that we don't always know in advance the length of the array we need.