



Strings, Arrays and Slices

This lesson is a flashback to the standard operations and their syntaxes defined on strings, arrays, and slices.

WE'LL COVER THE FOLLOWING



-  Useful code snippets for strings
 - Changing a character in a string
 - Taking a part (substring) of a string
 - Looping over a string with `for` or `for-range`
 - Finding number of bytes and characters in string
 - Concatenating strings
-  Useful code snippets for arrays and slices
 - Creation
 - Initialization
 - Cutting the last element of an array or slice line
 - Looping over an array (or slice) `arr` with `for` or `for-range`

Useful code snippets for strings

Changing a character in a string

Strings are immutable, so in fact a new string is created here.

```
str := "hello"  
c := []rune(s)  
c[0] = 'c'  
s2 := string(c) // s2 == "cello"
```

Taking a part (substring) of a string

```
substr := str[n:m]
```

Looping over a string with `for` or `for-range`

```
// gives only the bytes:
for i:=0; i < len(str); i++ {
    ... = str[i]
}

// gives the Unicode characters:
for ix, ch := range str {
    ...
}
```

Finding number of bytes and characters in string

Number of bytes in a string `str`:

```
len(str)
```

Number of characters in a string `str`:

The fastest way is:

```
utf8.RuneCountInString(str)
```

An equivalent way is:

```
len([]int(str))
```

Concatenating strings

The fastest way is:

```
// with a bytes.Buffer
var buffer bytes.Buffer
var s string
buffer.WriteString(s)
fmt.Print(buffer.String(), "\n")
```

Other ways are:

```
Strings.Join() // using Join function
str1 += str2 // using += operator
```



Creation

To create an array:

```
arr1 := new([len]type)
```

To create a slice:

```
slice1 := make([]type, len)
```

Initialization

To initialize an array:

```
arr1 := [...]type{i1, i2, i3, i4, i5}  
arrKeyValue := [len]type{i1: val1, i2: val2}
```

To initialize a slice:

```
var slice1 []type = arr1[start:end]
```

Cutting the last element of an array or slice line

```
line = line[:len(line)-1]
```

Looping over an array (or slice) arr with **for** or **for-range** #

```
for i:=0; i < len(arr); i++ {  
    ... = arr[i]  
}  
  
for ix, value := range arr {  
    ...  
}
```

This pretty much summarizes arrays, strings, and slices. The next lesson deals with structs, interfaces, and maps.

