Arry and Slices

Array

An array can be declared with the following syntax:

```
var <arr_name> [length]type
```

For example if I want to create a list of array with length of 5:

```
var a [5]int
```

The array is initialized with zero value of the type. So since it is integer it will be initialized to be 0. If it were `bool` it will all be `false`.

Indexing

Indexing the array is just as the same as with other array

```
arr_name[index]
```

There are no negative indexing like Python to prevent unintentional bugs.

Slices

Array is fine on it's own, but it comes with a down side, in the sense that it is very rigid. If you want to pass an array to a function, the size must be specified, otherwise, it cannot be passed.

Therefore, slices which are references to array are created to combat that issue. Passing slices of array allows the size to vary depending on the array.

The function below can only take in an array of 3 elements, if you want this function to work with any array, then you will have to make it to take in a slice.

```
func printArray(nums [3]int) {
	i := 0
	for i < len(nums) {
		fmt.Printf("The number is %d\n", nums[i])
		i += 1
```

```
    }
}
```

This variation which just took out the number becomes a slice, which will allow it to take in any slice of varying sizes.

```
func printArray(nums []int) {
    i := 0
    for i < len(nums) {
        fmt.Printf("The number is %d\n", nums[i])
        i += 1
    }
}
```

Revision #1
Created 2023-07-02 00:57:50 UTC by Tamarine
Updated 2023-07-02 01:43:26 UTC by Tamarine