### **Functions and Return values**

This lesson discusses functions, return values, named results and signature

#### WE'LL COVER THE FOLLOWING ^

- Functions
- Return values
  - Named Results

# Functions #

When declaring functions, the type comes after the variable *name* in the inputs.

The return type(s) are then specified after the function name and inputs, before writing the definition. Functions can be defined to return any number of values and each of them are put here.

Given below is an example function definition:

```
Environment Variables

Key: Value:

GOPATH /go

package main import "fmt"

func add(x int, y int) int {
    return x + y
}

func main() {
    fmt.Println(add(42, 13))
}
```







In the following example, instead of declaring the type of each parameter, we only declare one type that applies to both.



# Return values #

In the following example, the **location** function returns two string values.

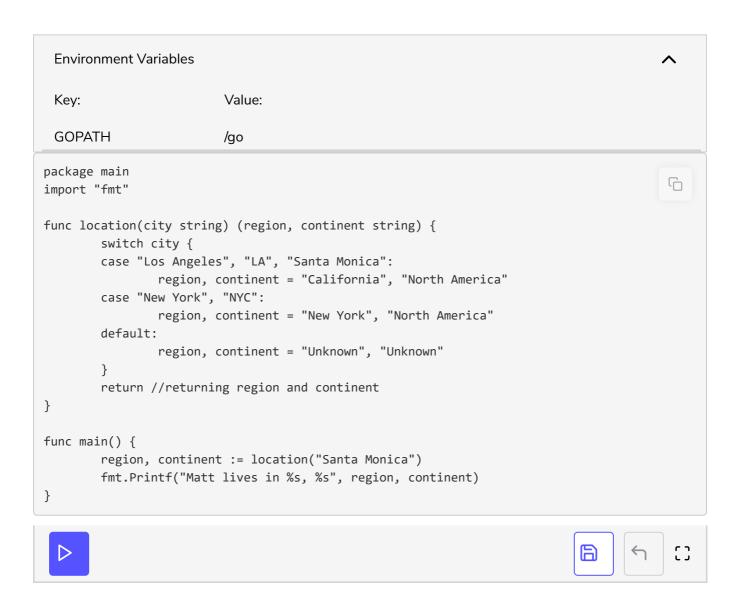


```
func main() {
    region, continent := location("Santa Monica")
    fmt.Printf("Matt lives in %s, %s", region, continent)
}
```

#### Named Results #

Functions take parameters. In Go, functions can return multiple "result parameters", not just a single value. They can be named and act just like variables.

If the result parameters are named, a return statement without arguments returns the current values of the results.



I personally recommend against using *named* return parameters because they often cause more confusion than they save time or help clarify your code.

In the next lesson, we will discuss <i>pointers</i> . Read on to find out more!			