codecademy

Learn Go: Functions

Go Pass by Value Parameter

When a Go function parameter is passed by value, it means only a copy of the value is accessed and manipulated inside the function. The original value of the variable that is passed as an argument to the function remains intact.

```
func makeMeOlder(age int) {
  age += 5
}

func main() {
  myAge := 10
  makeMeOlder(myAge)
  fmt.Println(myAge)
  // myAge is still 10
}
```

Go Variable Address

A Go variable occupies a slot in virtual memory and is accessible via an *address*. To access the address of a variable, type `&` followed by the variable name. The value of a variable address is in the form of a hexadecimal number, such as `0x414020`.

```
name := "Codecademy"
fmt.Printf("Address of %v is
%v", name, &name)
// Address of Codecademy is
0xc000010210
```

Go Pointer Dereferencing

The `*` operator preceding a data type is describing the data type for a Go pointer. For example:

```
var pointerToInt *int
// a pointer to a variable of
type int
```

The `*` operator preceding a variable is used to *dereference* a pointer variable. For example, the pointer variable, `x`, is assigned the address of variable, `y`. We dereference `x` by typing `*x`. By doing so, we can access and change the value of `y`. For example:

```
y := 3
var x *int = &y
*x = 5
fmt.Println(y)
// y is now 5
```