Structs & Custom Types

Structs → wrapper → custom datatypes ~

- Why Structs → Grouping , Reusability.
- What are Structs →
- Creating Structs →
- Adding methods to structs

```
type <s1> struct {         s2 {
    int
    float
    s2
}                          }
```

Structs

Why

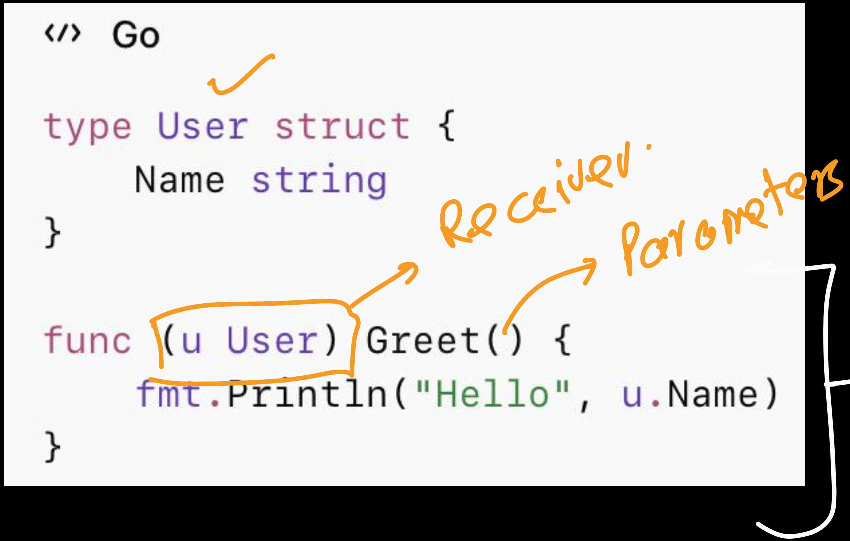
What is structs?

- A struct is a collection of fields.
- Struct fields are accessed using a dot

Functions vs Methods

Function: A function is a standalone block of code not attached to any type

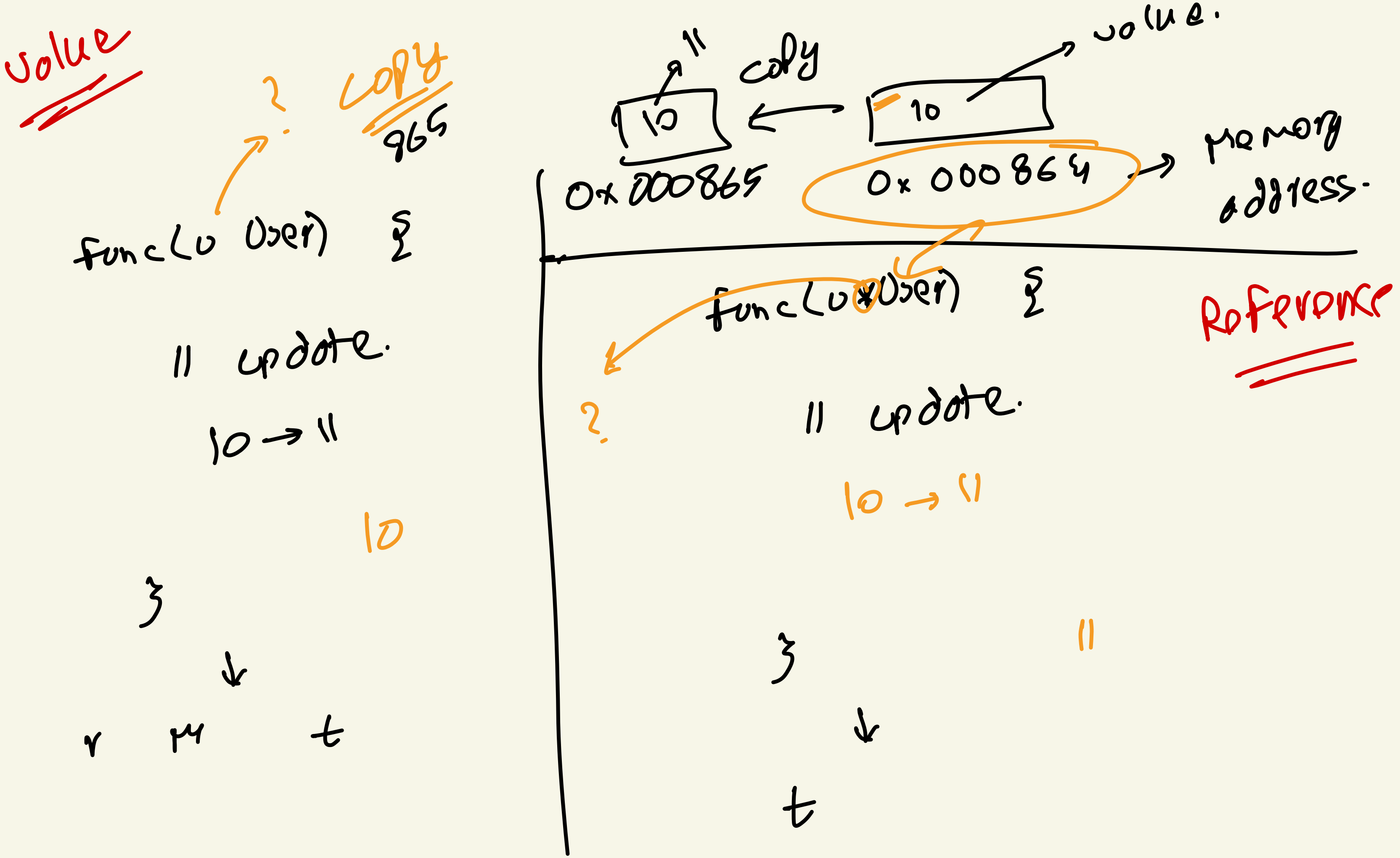
Method: A method is a function **attached to a struct** (or type) **via receiver**



Go

```go
type User struct {
    Name string
}

func (u User) Greet() {
    fmt.Println("Hello", u.Name)
}
```

F | M ✓



value
?
copy
965
func(u User) {
// update
10 → 11
10
}
r m t
11
copy
10
10
value.
0x000865
0x000864
memory address.
func(u *User) {
?
// update
10 → 11
11
}
t
Reference

Pointers

Pointers

* → Pointer → value of

& → Address → address of

```
var age int
age = 42
fmt.Println(age)
```

int

```
var pointer *int
fmt.Println(&age)
```

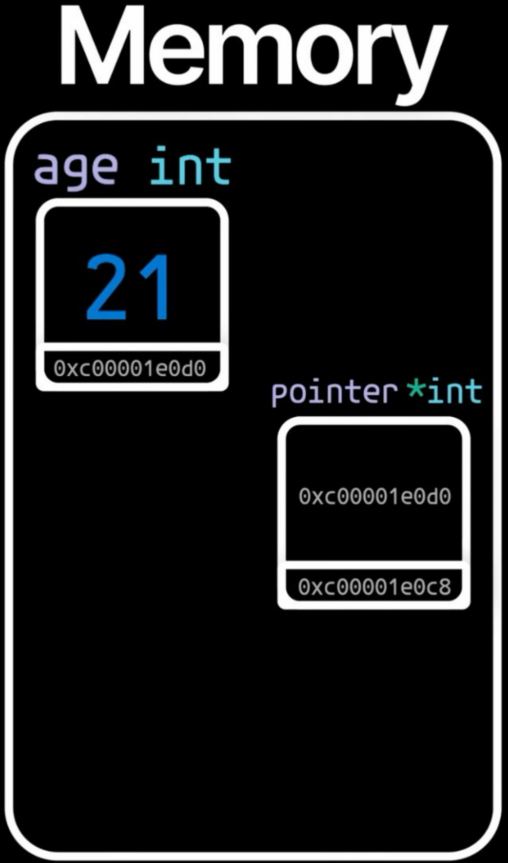
integer Pointer

→ address of

```
pointer =  &age
fmt.Println( *pointer )
```

→ value.

```
*pointer = 21
fmt.Println(age)
```



Pointers

Why Pointers?

- Avoid unnecessary copies (large copies)
- Directly **mutate values**
- Less code

What is a pointer?

- A pointer is a variable that stores the memory address of another variable, not the value itself

Pointers

Declaration of pointer

Dereferencing pointer

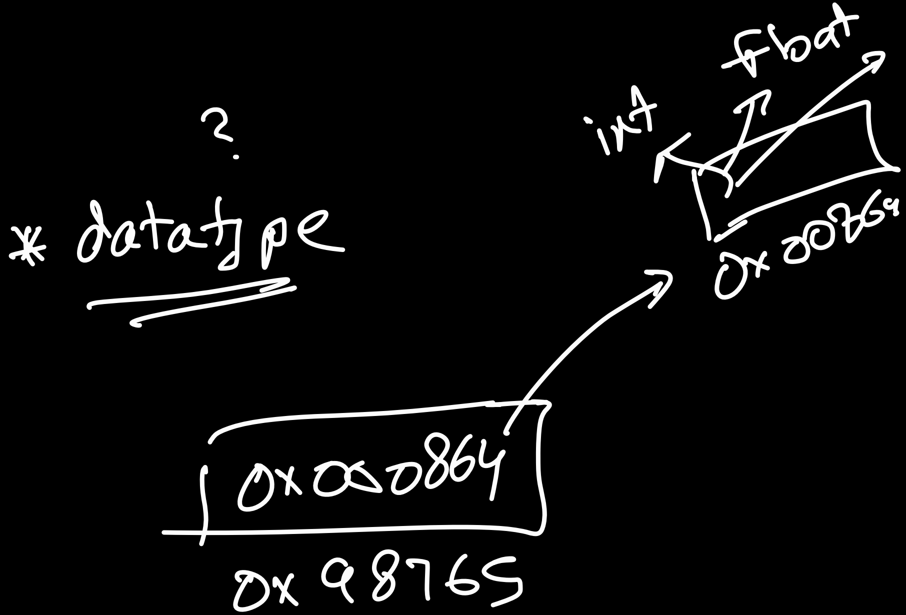
&

*

```
var marks int = 100

1  var markPoint *int

   markPointer = &marks

2  Print(*markPoint)
```

Interview Questions

1. How to declare pointer?
2. How to store value in pointer
3. How to perform Dereferencing?
4. Meaning of & symbol?
5. Meaning of * symbol?
6. What is default value of pointer?

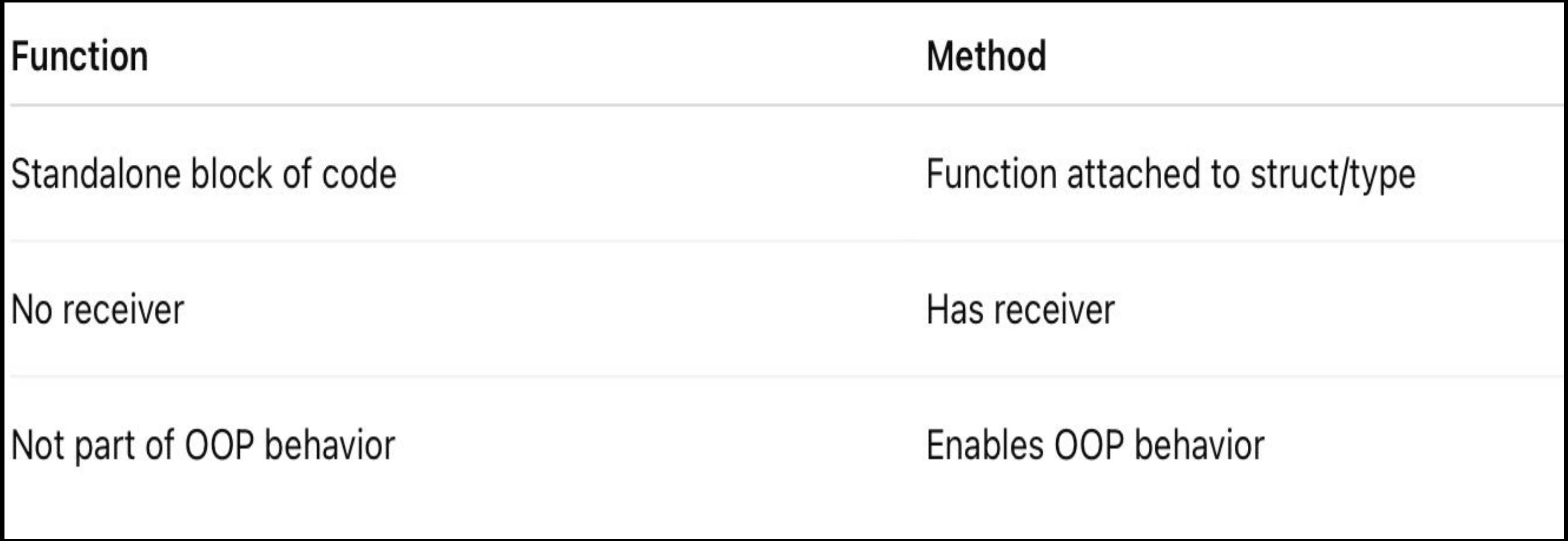


Interview Question

1. How Go passes arguments?

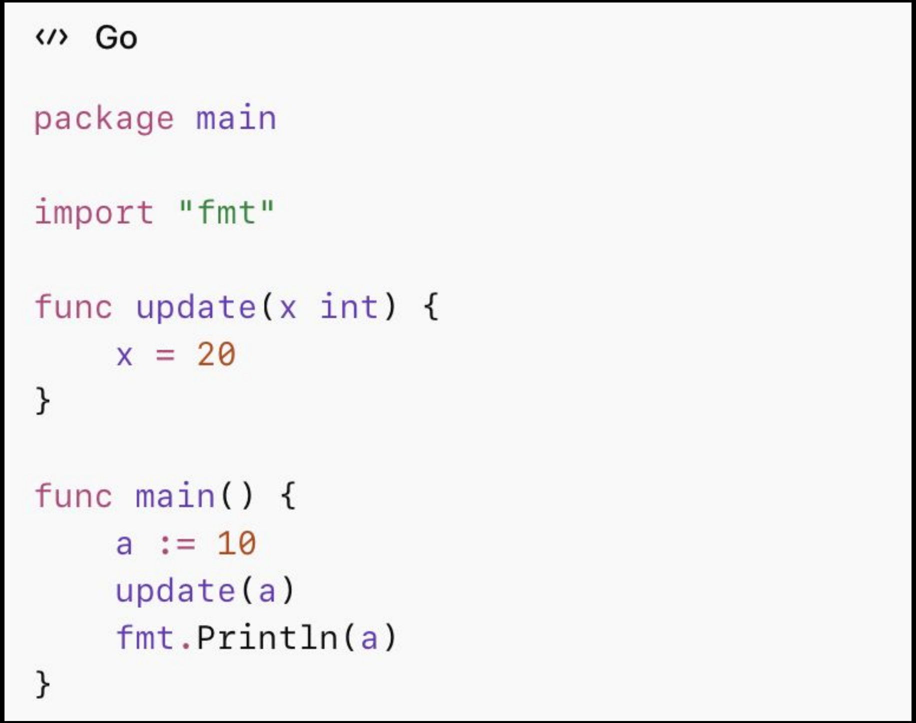
	✓ value	Reference
→	copy	×
→	mutate ×	✓

Functions vs Methods



Function	Method
Standalone block of code	Function attached to struct/type
No receiver	Has receiver
Not part of OOP behavior	Enables OOP behavior

Practice Question



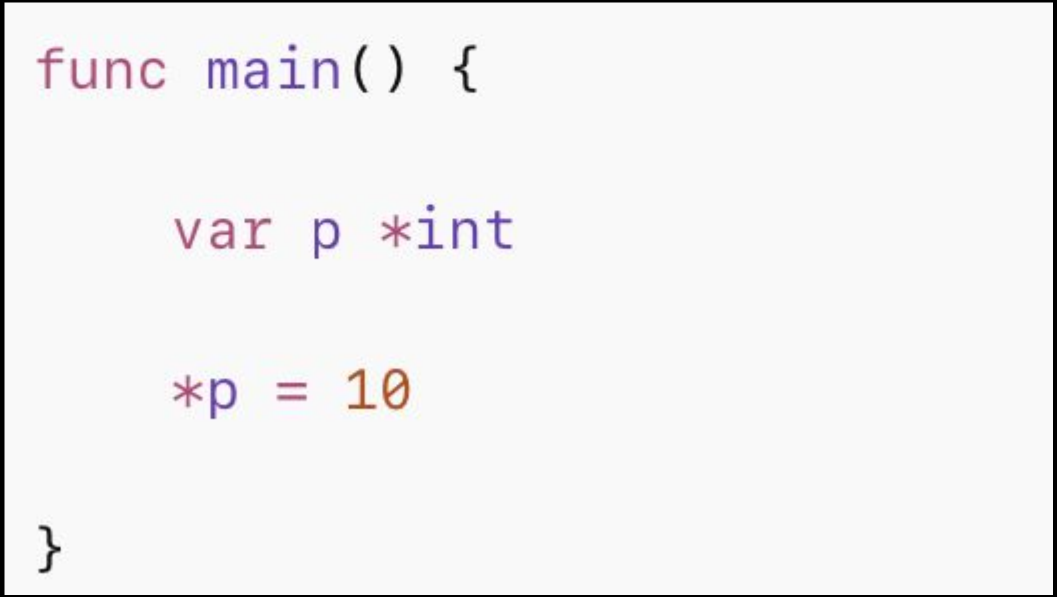
Go

```go
package main

import "fmt"

func update(x int) {
    x = 20
}

func main() {
    a := 10
    update(a)
    fmt.Println(a)
}
```

Interview Question

```
package main

import "fmt"

func main() {
    x := 5       → v Decl
    p := &x      → memory
    pp := &p     → memory    addr.

    (*(*pp)) = 50    →

    fmt.Println(x)
}
```

Interview Question



```go
func main() {

    var p *int

    *p = 10

}
```

Struct Contd

- Constructor Function
  - *controlled initialization*
  - *validation*
- Struct tags
- Struct embedding
- Composition