Go Struct

Go struct can be used to create user-defined types.

Struct is a composite type it means it can have different properties and each property can have their own type and value.

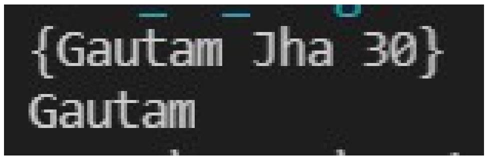
Struct can represent real-world entity with these properties.

We can access a property data as a single entity.

It is also valued types and can be constructed with the new() function.

```go
package main

import "fmt"

type person struct {
    FirstName string
    LastName  string
    age       int
}

func main() {
    x := person{
        age:       30,
        FirstName: "Gautam",
        LastName:  "Jha",
    }

    fmt.Println(x)
    fmt.Println(x.FirstName)
}
```

Output :-



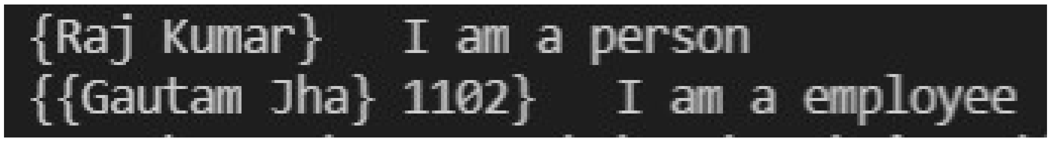
```
{Gautam Jha 30}
Gautam
```

Go Embedded Struct

Struct is a data type and can be used as an anonymous field (having only the type). One struct can be inserted or “embedded” into other struct.

It is a simple ‘inheritance’ which can be used to implement implementation from other type or types.

```go
package main

import "fmt"

type person struct {
    fname string
    lname string
}

type employee struct {
    person
    empld int
}

func (p person) details() {
    fmt.Println(p, " "+" I am a person")
}
func (e employee) details() {
    fmt.Println(e, " "+" I am a employee")
}

func main() {
    p1 := person{"Raj", "Kumar"}
    p1.details()
    e1 := employee{person: person{"Gautam", "Jha"},
empld: 1102}

    e1.details()
}
```



```
{Raj Kumar}   I am a person
{{Gautam Jha} 1102}   I am a employee
```