**Introduction:**

func main(){

slice\_val :=[]string {"one", "two", "three", "four", "five", "six"}

map\_val := map [string] int64{}

for itr:=0; itr<len(slice\_val); itr++{

map\_val[slice\_val[itr]] = 0

}

}

**Type:**

package main

import ("fmt"

"reflect"

)

func main(){

type student struct{

name string

age int

}

stu := student{"Akansha", 17}

fmt.Println(stu)

stu\_type := reflect.TypeOf(stu)

fmt.Println("type of 'stu' variable:", stu\_type)

fmt.Println("type of 'stu\_type' variable:", reflect.TypeOf(stu\_type))

fmt.Println("name of 'stu' variable type:", stu\_type.Name())

}

**Value:**

package main

import ("fmt"

"reflect"

)

func main(){

type student struct{

name string

age int

}

stu := student{"Akansha", 17}

fmt.Println(stu)

stu\_type := reflect.TypeOf(stu)

fmt.Println("value of 'stu' variable:", reflect.ValueOf(stu))

fmt.Println("value of 'stu\_type' variable:", reflect.ValueOf(stu\_type))

}

**Kind:**

package main

import ("fmt"

"reflect"

)

func main(){

type student struct{

name string

age int

}

stu := student{"Akansha", 17}

fmt.Println(stu)

stu\_type := reflect.TypeOf(stu)

stu\_kind := stu\_type.Kind()

fmt.Println("type of 'stu' variable:", stu\_type.Name())

fmt.Println("kind of 'stu\_type' variable:", stu\_kind)

}

**NumField:**

package main

import ("fmt"

"reflect"

)

func main(){

type student struct{

name string

age int

}

stu := student{"Akansha", 17}

stu\_value := reflect.ValueOf(stu)

num\_fields := stu\_value.NumField()

// fmt.Println("number of fields in 'stu':", stu.NumField()) // error "type student has no field or method NumField"

fmt.Println("number of fields in 'stu':", num\_fields, reflect.TypeOf(stu\_value))

stu\_type := reflect.TypeOf(stu)

fmt.Println("number of fields in 'stu':", num\_fields, reflect.TypeOf(stu\_type))

}

**Field:**

package main

import ("fmt"

"reflect"

)

func main(){

type student struct{

name string

age int

}

stu := student{"Akansha", 17}

stu\_value := reflect.ValueOf(stu)

num\_fields := stu\_value.NumField()

for itr:=0; itr<num\_fields; itr++{

fmt.Printf("value of %dth field is:%v of type %T\n", itr+1, stu\_value.Field(itr), stu\_value.Field(itr))

}

}

**Strings(), Int() & Float():**

package main

import ("fmt"

"reflect"

)

func main(){

type student struct{

name string

age int

income float32

}

stu := student{"Akansha", 17, 40000.565}

stu\_value := reflect.ValueOf(stu)

fmt.Printf("Type of name is %T and value is %v\n", stu\_value.Field(0), stu\_value.Field(0))

fmt.Printf("Type of age is %T and value is %v\n", stu\_value.Field(1), stu\_value.Field(1))

fmt.Printf("Type of income is %T and value is %v\n", stu\_value.Field(2), stu\_value.Field(2))

name := reflect.ValueOf(stu.name).String()

age := reflect.ValueOf(stu.age).Int()

income := reflect.ValueOf(stu.income).Float()

fmt.Printf("Type of name is %T and value is %v\n", name, name)

fmt.Printf("Type of age is %T and value is %v\n", age, age)

fmt.Printf("Type of income is %T and value is %v\n", income, income)

}

**Laws of reflection:**

**Rule1:**

package main

import ("fmt"

"reflect"

)

func show\_value(i interface{}){

value\_type := reflect.TypeOf(i)

value := reflect.ValueOf(i)

fmt.Printf("Passed value is %v of type %v\n", value, value\_type.Kind())

fmt.Println(reflect.TypeOf(value))

fmt.Println(reflect.TypeOf(value\_type))

}

func main(){

type student struct{

name string

age int

income float32

}

stu := student{"Akansha", 17, 40000.565}

show\_value(stu)

}

**Rule2:**

package main

import ("fmt"

"reflect"

)

func show\_value(i interface{}){

value\_type := reflect.TypeOf(i)

value := reflect.ValueOf(i)

fmt.Println(reflect.TypeOf(value))

fmt.Println(reflect.TypeOf(value\_type))

interface\_val := value.Interface()

fmt.Println("Interface value is:", interface\_val)

fmt.Println("Interface value type is:", reflect.TypeOf(interface\_val).Kind())

}

func main(){

type student struct{

name string

age int

income float32

}

stu := student{"Akansha", 17, 40000.565}

show\_value(stu)

}

**Rule3:**

package main

import ("fmt"

"reflect"

)

func main(){

var str string = "Hello"

val := reflect.ValueOf(str)

fmt.Println(val)

val.SetString(“World”) // will throw panic

}

package main

import ("fmt"

"reflect"

)

func main(){

var str string = "Hello"

val := reflect.ValueOf(str)

fmt.Println(val)

fmt.Println(val.CanSet())

}