Go Training

Session 9

Assignment Solution

1. Write a program to output each character of the string “ Hellő There ”

```go
package main
import "fmt"


func main() {

    str := "Hellő There"

    for _, char := range str{
        fmt.Println(string(char))
    }
}
```

```go
package main
import "fmt"


func main() {

    str := "Hellő There"
    data := []rune(str)

    for i:=0;i<len(data);i++{
        fmt.Println(string((data[i])))
    }
}
```

Assignment Solution

1. Write a program to implement a function uniqueFunc() which takes variable number of arguments and types of arguments can also vary. It should print square for int argument, perimeter for rectangle struct argument and some error message for other types.

   uniqueFunc(3, 4, rectangle{2,5}) => 9, 16, 14

```go
package main
import "fmt"

type rect struct{
    len int
    wid int
}

func (r rect) perimeter() int{
    return 2*(r.len + r.wid)
}

func uniqueFunc(i ...interface{}){
    for _,val := range i{
        switch val.(type){
        case rect:
            fmt.Println(val.(rect).perimeter())
        case int:
            fmt.Println(val.(int)*val.(int))
        default:
            fmt.Println("type not compatible")
        }
    }
}

func main() {

    uniqueFunc(3,rect{3,4}, 5)
}
```

error

- An error is just a value that a function can return if something unexpected happened
- Zero value of type error is nil

```go
package main
import "fmt"
import "os"

func main() {

    data, err := os.Open("testFile.go")
    if(err!=nil){
        fmt.Println(err)
    } else{
        fmt.Printf("data type is %T", data)
    }
}

```

error

```go
package main
import "fmt"
import "errors"

func main() {

    val,err := myFunc(2,-3)
    if(err!=nil){
        fmt.Println(err)
    } else{
        fmt.Println(val)
    }
}

func myFunc(x,y int) (int,error) {
    var err error
    if(x*y < 0){
        err = errors.New("you entered a negative number")
    }

    return x*y, err
}
```

error

```go
package main
import "fmt"

// type error interface {
//  Error() string
// }

type errorString struct {
    str string
}

func (es *errorString) Error() string {
    return es.str
}

func NewErr(txt string) error {
    return &errorString{"error is " + txt}
}

func main() {

    var err error
    err = NewErr("check")
    fmt.Println(err)
}
```

Thank You