**Introduction:**

package main

import ("fmt"

"os"

)

func main(){

file, err := os.Open("./example.json")

fmt.Println("file status:", file)

fmt.Println("error status:", err)

}

**Error type:**

package main

import "fmt"

type My\_error struct{}

func (error\_object \*My\_error) Error() string{

return "Unexpected outcome"

}

func main(){

fmt.Println(&My\_error{})

}

package main

import "fmt"

type My\_error struct{}

func (error\_object \*My\_error) Error() string{

return "Unexpected outcome"

}

func main(){

obj := My\_error{}

fmt.Println(obj.Error())

}

**Error package:**

package main

import ("fmt"

"os"

"errors"

)

func main(){

file, err := os.Open("./example.json")

if(err!=nil){

error\_msg := errors.New("file operation failed")

fmt.Println(error\_msg)

}

if(err==nil){

fmt.Println("file opened successfully", file)

}

}

**Defer statement:**

package main

import ("fmt"

"os"

)

func open\_file(){

defer file.Close()

file, err := os.Open("./example.txt")

fmt.Println("file status:", file)

fmt.Println("error status:", err)

}

func main(){

open\_file()

}

package main

import ("fmt"

"os"

)

func open\_file(){

file, err := os.Open("./example.txt")

defer file.Close()

fmt.Println("file status:", file)

fmt.Println("error status:", err)

}

func main(){

open\_file()

}

package main

import "fmt"

func show\_value(){

for itr:=0; itr<=10; itr++{

defer fmt.Println("value:", itr)

}

}

func main(){

show\_value()

}

package main

import "fmt"

func update\_return\_value() (i int) {

defer func() { i = i+10 }()

return 1

}

func main(){

fmt.Println("updated return value:", update\_return\_value())

}

**Panic:**

package main

import "fmt"

var result int

func op(a, b int) int {

result = a / b

return result

}

func main(){

result = op(10, 0)

fmt.Println("Completed")

}

package main

import "fmt"

var result int

func op(a, b int) int {

if(b==0){

panic("Can't divide a number by zero!")

}

result = a/b

return result

}

func main(){

result = op(10, 0)

fmt.Println("Completed")

}

package main

import "fmt"

func show\_value(max int){

for itr:=1; itr<=max; itr++{

if(itr==5){

panic("Not acceptable")

}

fmt.Println(itr)

}

}

func main(){

show\_value(10)

fmt.Println("Completed")

}

package main

import "fmt"

var result int

func op\_defer(){

fmt.Println("op defer function executed")

}

func main\_defer(){

fmt.Println("main defer function executed")

}

func op(a, b int) int {

defer op\_defer()

if(b==0){

panic("Can't divide a number by zero!")

}

result = a/b

return result

}

func main(){

defer main\_defer()

result = op(10, 0)

fmt.Println("Completed")

}

**Recover:**

package main

import "fmt"

var result int

func op\_defer(){

fmt.Println("op defer function executed")

}

func main\_defer(){

fmt.Println("main defer function executed")

}

func op(a, b int) int {

defer func() {

if r := recover(); r != nil {

fmt.Println("Panic recovered", r)

op\_defer()

}

}()

if(b==0){

panic("Can't divide a number by zero!")

}

result = a/b

return result

}

func main(){

defer main\_defer()

result = op(10, 0)

fmt.Println("Completed")

}