**Introduction:**

func main() {

// str3 := "世Hello Go界"

var str string

fmt.Printf("By default strings are:%q\n", str)

str = "Hello Go"

fmt.Println("After assigning a value string is:", str)

str\_new := "Hello world"

fmt.Printf("After shorthand declaration, double quoted value of str\_new is:%q\n", str\_new)

fmt.Printf("Uninterpreted bytes of str\_new is:%s\n", str\_new)

fmt.Printf("base16 notation of str=%x", str)

}

**Length of string:**

func main() {

var str string

str = "Hello Go"

fmt.Println("Length of string 'str' is:", len(str))

str\_new := "世Hello Go世"

fmt.Printf("Length of string 'str\_new' is:", len(str\_new))

}

**Iterate through a string:**

func main() {

var str string

str = "Hello Go"

fmt.Println("characters of str")

for itr:=0; itr<len(str); itr++{

fmt.Printf("%c ", str[itr])

}

fmt.Println("\nUTF-8 encoded values of str")

for itr:=0; itr<len(str); itr++{

fmt.Printf("%x ", str[itr])

}

str\_new := "世Hello Go世"

fmt.Println("\ncharacters of str\_new")

for itr:=0; itr<len(str\_new); itr++{

fmt.Printf("%c ", str\_new[itr])

}

fmt.Println("\nUTF-8 encoded values of str\_new")

for itr:=0; itr<len(str\_new); itr++{

fmt.Printf("%x ", str\_new[itr])

}

}

**Rune:**

func main() {

str\_new := "世Hello Go世"

rune\_slice := []rune(str\_new)

fmt.Println("\ncharacters of str\_new")

for itr:=0; itr<len(rune\_slice); itr++{

fmt.Printf("%c ", rune\_slice[itr])

}

fmt.Println("\nUTF-8 encoded values of str\_new")

for itr:=0; itr<len(str\_new); itr++{

fmt.Printf("%x ", str\_new[itr])

}

}

**Iterate using range keyword:**

func main() {

str\_new := "世Hello Go世"

fmt.Println("\nCharacters of str\_new")

for \_, item := range str\_new{

fmt.Printf("%c ", item)

}

}

**Multiline string:**

str := "Hello

Go"

fmt.Printf("string with double quotes:%s\n", str) // error

func main() {

str := "Hello\n Go"

fmt.Printf("String with double quotes:%s\n", str)

str\_new := `Hello\n Go`

fmt.Printf("String with backtick:%s\n", str\_new) //treats \n as normal character

str\_new = `Hello

Go`

fmt.Printf("Linebreak in string with backtick:%s\n", str\_new)

}

**Convert slice of runes to string:**

func main() {

rune := [] rune{0x4e16, 0x0048, 0x0065, 0x006c, 0x006c, 0x006f, 0x0047, 0x006f, 0x4e16}

str := string(rune)

fmt.Println("string from slice of rune:", str)

}

**Convert slice of bytes to string:**

func main() {

rune := [] byte{228, 184, 150, 72, 101, 108, 108, 111, 71, 111, 228, 184, 150}

str := string(rune)

fmt.Println("String from a slice of bytes:", str)

}

**Immutable nature of strings:**

func main() {

str := "Hello Go"

fmt.Printf("%c", str[3]) // l

str[3] = 'y' // cannot assign to str[3]

}

func main() {

str := "Hello Go"

fmt.Printf("Character at third index: %c\n", str[3])

rune := []rune(str)

rune[3] = 'M'

str = string(rune)

fmt.Printf("Character at third index after changing it's value: %c", str[3])

}

**Character comparison:**

func main() {

if('a'>'A'){

fmt. Println("a is greater tha A")

}

fmt. Println("a is greater than z:", 'a'>'z')

for itr:='a'; itr<'m'; itr++{

fmt.Print(itr)

fmt.Printf(" "+string(itr)+"\n")

}

}

**Converting string in uppercase and lowercase:**

package main

import (

"fmt"

"strings"

)

func main() {

fp := fmt.Println

str := "Hello Go"

fp("string in uppercase:", strings.ToUpper(str)) // string in uppercase: HELLO GO

fp("string in lowercase:", strings.ToLower(str)) // string in lowercase: hello go

}

**Search for string prefix/ suffix:**

func main() {

fp := fmt.Println

str := "Hello Go"

fp("has prefix He", strings.HasPrefix(str, "He")) // has prefix He true

fp("has prefix he", strings.HasPrefix(str, "he")) // has prefix he false

fp("has prefix go", strings.HasSuffix(str, "go")) // has prefix go false

fp("has prefix Go", strings.HasSuffix(str, "Go")) // has prefix Go true

}

**Search for specific text:**

func main() {

fp := fmt.Println

str := "Hello Go"

fp("contains ellO", strings.Contains(str, "ellO")) // contains ellO false

fp("contains o G:", strings.Contains(str, "o G")) // contains o G: true

fp("containsany welcome:", strings.ContainsAny(str, "welcome")) // containsany welcome: true

fp("containsany shubh:", strings.ContainsAny(str, "shubh")) // containsany shubh: false

}

**Number of occurrences in a string:**

func main() {

fp := fmt.Println

str := "Grow And Grow With Golang"

fp("occurences of Gr:", strings.Count(str, "Gr")) // occurences of Gr: 2

fp("occurences of welcome:", strings.Count(str, "welcome")) // occurences of welcome: 0

}

**Find position of a string:**

func main() {

fp := fmt.Println

str := "Grow And Grow With Golang"

fp("position of Gr:", strings.Index(str, "Gr")) // position of Gr: 0

fp("position of welcome:", strings.Index(str, "welcome")) // position of welcome: -1

}

**Find position of unicode codepoint:**

func main() {

fp := fmt.Println

str := "Grow And Grow With Golang"

fp("position of first appearing Unicode codepoint in Ground:", strings.IndexAny(str, "Ground")) // position of first appearing Unicode codepoint in Ground: 0

fp("position of first appearing Unicode codepoint in sub:", strings.IndexAny(str, "sub")) // position of first appearing Unicode codepoint in sub: -1

}

**Find position of string from last:**

func main() {

fp := fmt.Println

str := "Grow And Grow With Golang"

fp("last position of Gr:", strings.LastIndex(str, "Gr")) // last position of Gr: 9

fp("last position of welcome:", strings.LastIndex(str, "welcome")) // last position of welcome: -1

}

**Find position of Unicode codepoint from Last:**

func main() {

fp := fmt.Println

str := "Grow And Grow With Golang"

fp("position of last appearing Unicode codepoint in Ground: ", strings.LastIndexAny(str, "Ground”)) // position of last appearing Unicode codepoint in Ground: 23

fp("position of last appearing Unicode codepoint in sub: ",

strings.LastIndexAny(str, "sub")) // position of last appearing Unicode codepoint in sub: -1

}

**Convert array to string:**

func main() {

fp := fmt.Println

str := [] string {"my", "name", "is", "rahul"}

fp(strings.Join(str, "-")) // my-name-is-rahul

}

**Replacing specified string:**

func main() {

fp := fmt.Println

str := "my-name-is-rahul"

fp(strings.Replace(str, "-", " ", 0)) // my-name-is-rahul

fp(strings.Replace(str, "-", " ", 1)) // my name-is-rahul

fp(strings.Replace(str, "-", " ", 9)) // my name is rahul

fp(strings.Replace(str, "welcome", " ", 9)) // my-name-is-rahul

fp(strings.ReplaceAll(str, "-", " ")) // my name is rahul

}

**Convert string to slice:**

func main() {

fp := fmt.Println

str := "my-name-is-rahul"

fp(strings.Split(str, "-")) // [my name is rahul]

fp(strings.Split(str, "welcome")) // [my-name-is-rahul]

fp(strings.Split(str, "")) // [m y - n a m e - i s - r a h u l]

fp(strings.Fields("welcome \t in the world of \n Go")) // [welcome in the world of Go]

}

**Removing specific Unicode codepoint:**

func main() {

fp := fmt.Println

str := "-my-name-is-rahul-"

fp(strings.Trim(str, "-")) // my-name-is-rahul

fp(strings.TrimLeft(str, "-")) // my-name-is-rahul-

fp(strings.TrimRight(str, "-")) // -my-name-is-rahul

fp(strings.TrimFunc(str, func(c rune) bool {

return c >= 23 && c <= 50

})) // my-name-is-rahul

fp(strings.TrimLeftFunc(str, func(c rune) bool {

return c >= 23 && c <= 50

})) // my-name-is-rahul-

fp(strings.TrimRightFunc(str, func(c rune) bool {

return c >= 23 && c <= 50

})) // -my-name-is-rahul

fp(strings.TrimPrefix(str, "-my-")) // name-is-rahul-

fp(strings.TrimSuffix(str, "-rahul-")) // -my-name-is

fp(strings.TrimSuffix(str, "welcome")) // -my-name-is-rahul- (Unchanged)

str = " -my-name-is-rahul- "

fp(strings.TrimSpace(str)) // -my-name-is-rahul-

}