Hashmap

1. String Character count

func main(){

s:="hello world"

charCount:=make(map[byte]int)

for i:=0;i<len(s);i++{

charCount[s[i]]++

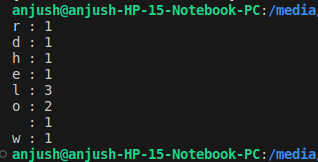
}

for k,v:= range charCount{

fmt.Printf("%c : %d\n",k,v)

}

}



1. Group array of words by length

func main(){

words:=[]string{"apple","banana","cat","dog","mango","japan"}

group:=GroupByLength(words)

for length,words:=range group{

fmt.Printf("length %d : %v \n",length,words)

}

}

func GroupByLength(words []string) map[int][]string{

group:=make(map[int][]string)

for \_,word:=range words {

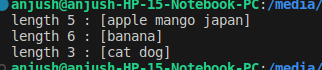
length:=len(word)

group[length]=append(group[length],word)

}

return group

}



1. Implementing set using hashmap

func main(){

s:=NewSet()

s.Add(1)

s.Add(2)

s.Add(4)

if s.Contains(3){

fmt.Println("3 conatins in the set")

}else{

fmt.Println("not contains")

}

}

type Set struct{

data map[int]struct{}

}

func NewSet() \*Set{

return &Set{data: make(map[int]struct{})}

}

func(s \*Set) Add(value int){

s.data[value]= struct{}{}

}

func (s \*Set) Contains(value int)bool{

\_,exists:=s.data[value]

return exists

}

