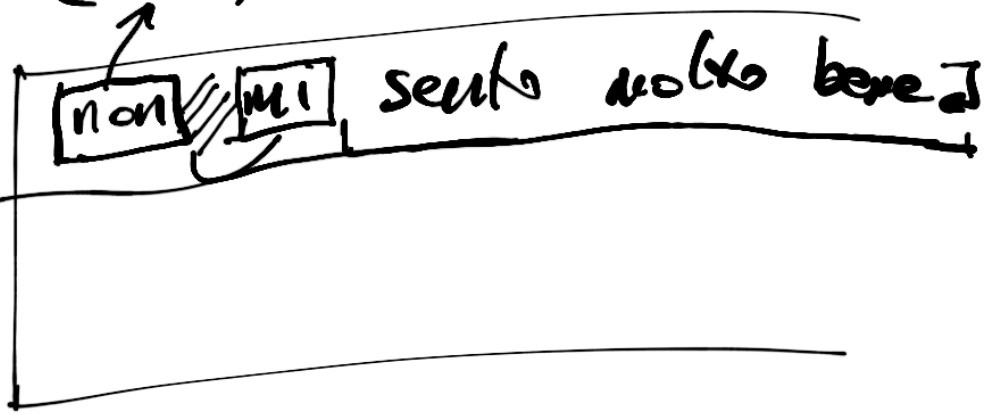
LETTURA DI STRINGHE

①

```
var s string
fmt.Scan(&s)
...
fmt.Scan(&t)
```



②

```
import (
        "bufio"
        "os"
)
```

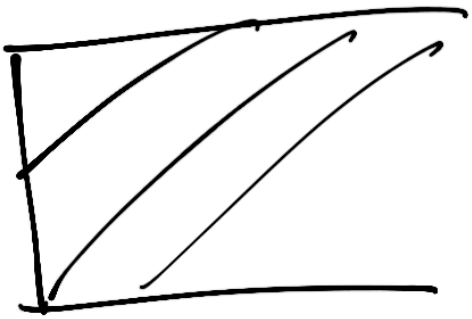
```
scanner := bufio.NewScanner(os.Stdin)
for scanner.Scan() {
        line := scanner.Text()

}
```

FUNZIONE CHE DATA UNA STRINGA BINARIA ('0', '1') LA CONVERTA IN UN int

COMMA-OK IDIOM

```
func binaryToInt(s string) (int, bool) {
    if len(s) == 0 {
        return 0, false
    }
    v := 0
    for _, r := range s {
        switch r {
        case '0':
            v *= 2
        case '1':
            v = 2*v + 1
        default:
            return 0, false
        }
    }
    return v, true
}
```

```
var a int
var ok bool
a, ok = binary.ToString(s)
if !ok {
    fmt.Println("String d
                non convertibile")
    a = 0
} else {
```



```
}
```

```
a, _ = binary.ToString(s)
_, ok = binary.ToString(s)
```

strconv

- Atoi (s string) (int, error)

```
n, err := strconv.Atoi(s)
if err != nil {
        fmt.Println("ERROR")
        ...
} else {
        ...use n...
}
```

```
n, _ := strconv.Atoi(s)
```

COMMA-ERROR IDIOM

```
func f(...) (..., error)
{
nil
```

Strconv

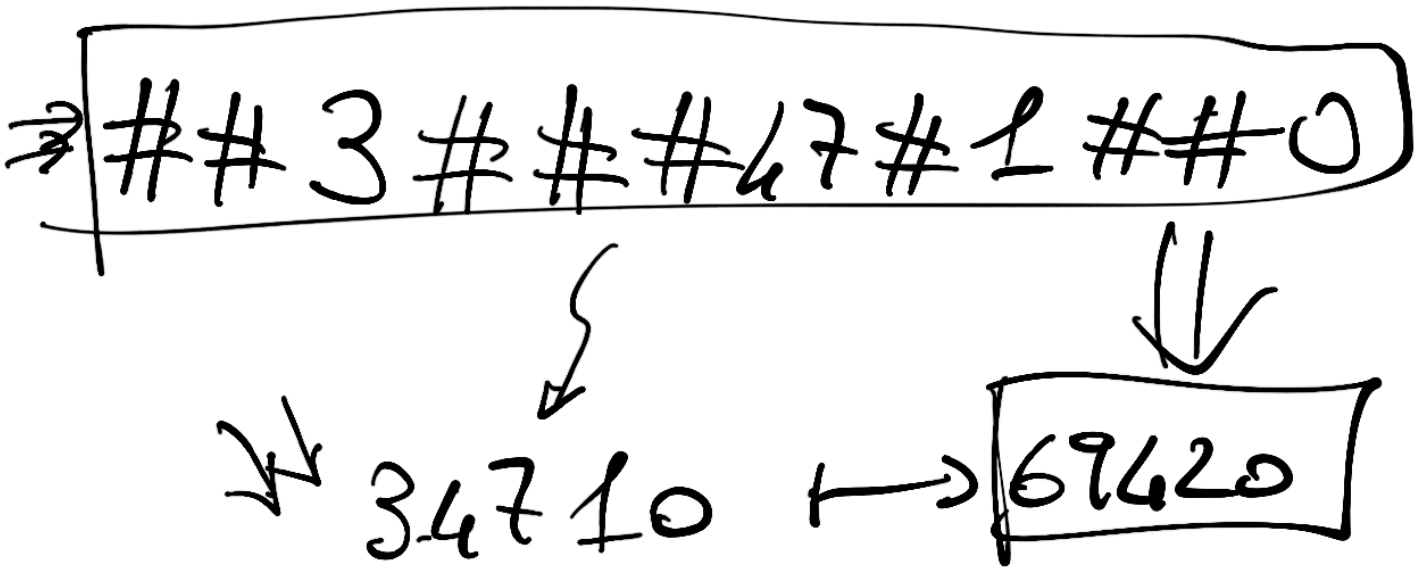
- Atoi (s string) (int, error)
- ParseInt (s string, base int, bitSize int) (int64, error)
- Itoa (x int) string

- ParseBool (s string) (bool, error)
  ParseFloat (s string) (float64, error)
- FormatBool (x bool) string
  FormatFloat64 (x float64) string

ESEMPIO

```
n, _ := ParseInt(s, 2, 16) ✓
value := int16(n)
```

```
go doc strconv

go doc strconv.Atoi
```



FUNZIONE CHE DATA UNA STRINGA COME QUELLA SOPRA RESTITUISCA IL DOPPIO DEL NUMERO MISTERIOSO

```
func misterioso(s string) (int, bool) {
    var pulita string
    for _, r := range s {                    ③
        if unicode.IsDigit(r) {
            pulita += string(r)
        } else if r != '#' {
            return 0, false
        }
    }
    n, err := strconv.Atoi(pulita)
    return 2*n, err == nil
}
```

---

① `r == '0' || r == '1' || r == '2' ...`

② `r >= '0' && r <= '9'`

FUNZIONE CHE DATO n RESTITUISCE LA STRINGA


```
func quadrato(n int) string {
    var s string
    for i:=0; i<n; i++ {
        s+= "*"
    }
    s+= "\n"
    for i:=0; i<n-2; i++ {
        s+= "*"
        for j:=0; j<n-2; j++ {
            s+= " "
        }
        s+= "*\n"
    }
}
```

```
func quadrato(n int) string {
    var s string
    for i:=0; i<n; i++ {
        s+="*"
    }                                   1^A RIGA
    s+="\n"
    for i:=0; i<n-2; i++ {
        s+="*"
        for j:=0; j<n-2; j++ {
            s+=" "
        }                               RIGHE INT.
        s+="*\n"
    }
    for i:=0; i<n; i++ {
        s+="*"
    }                                   ULTIMA RIGA
    s+="\n"
    return s
}
```

```
func rep(c rune, n int) string {
    var s string
    for i:=0; i<n; i++ {
        s += string(c)
    }
    return s
}
```

```
func quadrato(n int) string {
    var s string
    s = rep('*', n) + "\n"          // 1ª RIGA
    for i:=0; i<n-2; i++ {
        s += "*"
        s += rep(' ', n-2)
        s += "*\n"
    }
    return s + rep('*', n) + "\n"   // 2ª RIGA
}
```

strings

Contains (h, n string) bool
- restituisce true se n
  compare in h

Counts (h, n string) int
- restituisce quante volte
  n compare in h

HasPrefix (s, p string) bool
- restituisce true se p
  è un prefisso s

HasSuffix (s, x string) bool
- restituisce true se x
  è un suffisso s

ToLower (s string) string

Index(s, t string) int

- Restituisce la prima posizione (intera) di s a partire dalla quale si trova un'occorrenza di t (restituisce -1 se non ce ne sono)

```
          0 1 2 3 4
s = "ciao mamma ciao mamma"
          ↑
          5

t = "mamma"
```

SCRIVERE UNA FUNZIONE CHE DATE s E t STAMPI TUTTE LE POSIZ. DI s IN CUI t COMPARE