CONVERTITORE []rune

città☺ → UTF-8 byte

7 rune

```
var s string
s = "città☺"
r := []rune(s)
for i := len(r)-1; i>=0; i-- {
    fmt.Printf("%c", r[i])
}
fmt.Println()
```

MAPPE (dizionari, array associativo)

= insieme di coppie chiave/valore

Key →	value
"Pippo"	15
"giornni"	7
"carlo"	7
"ciao"	12

↑ distinte

map [TIPO CHIAVE] TIPO VALORE

restrizioni

var m1 map[string]int

var m2 map[rune] string

var m3 map[data] int

var m4 map[data] carta

USO

```
var m map[string] int
m = make(map[string] int)
```
→ creazione

```
m["pippo"] = 15
m["giovanni"] = 7
m["carlo"] = 7
m["ciao"] = 12
```
AGGIUNTE DI NUOVE COPPIE K/V o MODIFICA VALORI DI

len(m) ⇝ numero di coppie

```
m["pluto"] = 4
m["pippo"] = 3
```

INTERROGAZIONE (QUERY)

```
fmt.Println(m["pippo"])
```

- se la chiave "pippo" è presente, restituisce il valore associato
- altrimenti restituisce zero (del tipo valore)

QUERY COMPLETA

```
v, present := m["pippo"]
```

v → valore se c'è, altrimenti zero

present → boolean che dice se è presente

SCANSIONE (ordine → imprevedibile)

```
for k,v := range m1 {
    ...
}
```

CANCELLA UNA CHIAVE

```
delete(m1, "pippo")
```

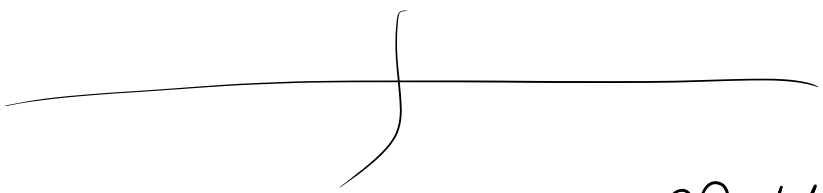
```
m1 := map[string]int{"ciao": 7, "mamma": 4, "domani": 12}
```

LETTERALE MAPPA

NUMERI PSEUDO CASUALI

```
import    math/rand
rand.Seed(time.Now().UTC().
          UnixNano)

rand.Intn(1000)
```



PREVEDERE = COMPRIMERE