rune

```
var x rune
x = 'A'
x = 98
x = '\u001A'
x = '\U001A1895'
fmt.Println(string(x))
```

```
if x >= 'a' && x <= 'z'
```

```
if (x >= 'a' && x <= 'z') ||
   (x >= 'A' && x <= 'Z')
```

```
if x >= 'a' && x <= 'z' {
    x = x - 'a' + 'A'
}
```

unicode.IsLetter (a rune) bool

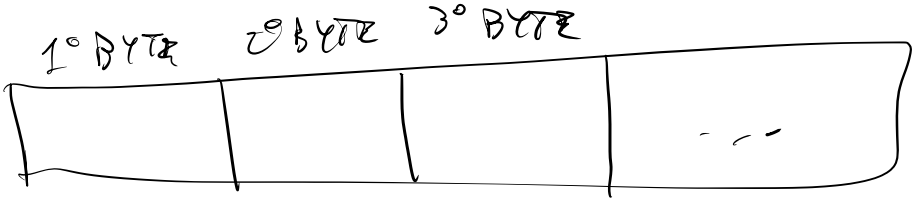
unicode.ToLower (a rune) rune

...

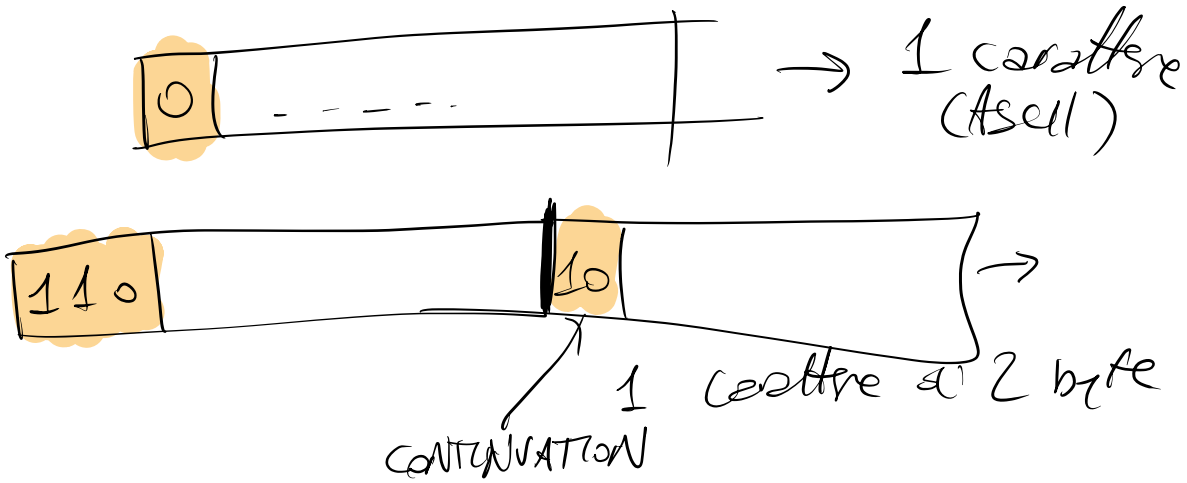
CODIFICA
UTF-8

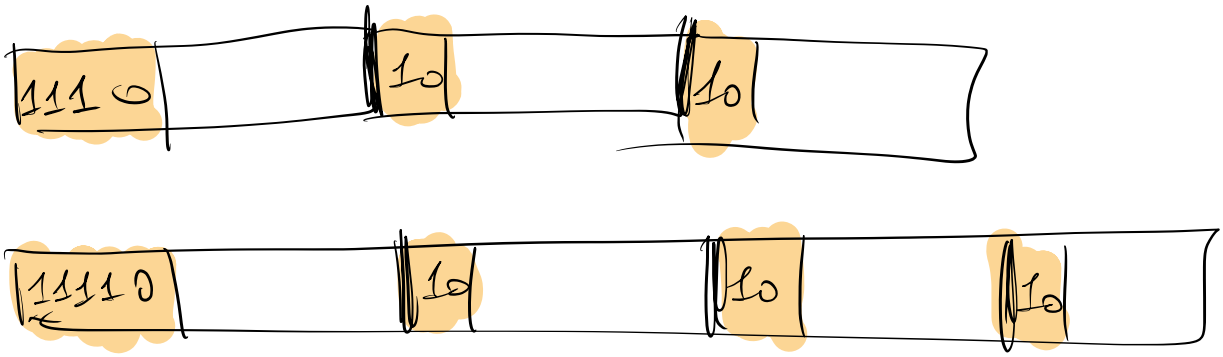
Paolo ␣ Baldi → UTF-32

UTF-8



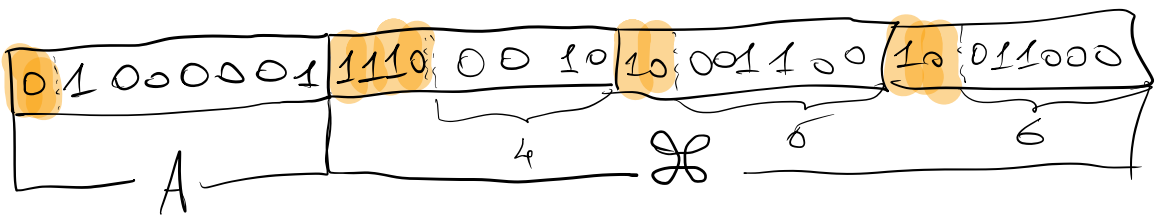
LEADING BYTE




1110
10
10
11110
10
10
10

"A⌘"

A	65	01000001
⌘	8984	1000 11000 11000



TIPO
string

var x string

≡ seq. di caratteri rappresentate in UTF-8

x = "A⌘"
x = "Paolo Boldi"
x = "Ciao \n mamma"

① len(x) → lunghezza in byte

② x + y → concatenazione

③ `x[i]` ⟶ i-esimo byte di x

④ `x[i:j]` ⟶ estrae la sottostringa dal byte i-esimo (compreso) al byte j-esimo (escluso)

`x[i:]`

`x[:j]`

⑤ `fmt.Scan(&x)`

CICLO for-RANGE

```
for i, r := range s {

}
```

INDICE (BYTE) — RUNE — STRINGA

DATA UNA STRINGA CONTARE LE LETTERE MINUSCOLE

```
var s string
fmt.Scan(&s)
count := 0
for _, c := range s {
    if unicode.IsLower(c) {
        count++
    }
}
fmt.Println(count)
```

```
for i := 0; i < len(s); i++ {
    if unicode.IsLower(rune(s[i])) {
        count++
    }
}
```

- DATA UNA URL TROVARE LO HOST (AUTHORITY)

https://fineco.it/conto/.....

- CONTARE IN UNA STRINGA IL NUMERO DI "PAROLE". (seq. caratteri separate da spazi)

"ciao Giovanni, ci vediamo domani"

```
var s string
s = "ciao Giovanni ci ----- "

numParole := 0
for i := 0; i < len(s)-1; i++ {
    if s[i] == ' ' && s[i+1] != ' ' {
        numParole++
    }
}
```