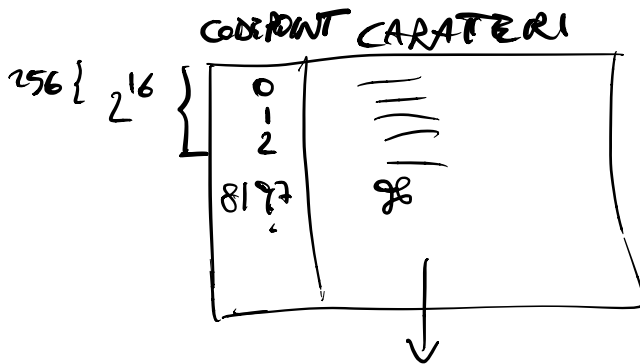


RUNE E CARATTERI

- Unicode



$$17 \cdot 2^{16}$$

PLANE
≈ 1'100'000



encoding
- UTF-32
- rune
- UTF-8

var c rune

c = 'A'

c = 17477 ^{code point} _{esad.}

c = '\Uxxxx xxxx'

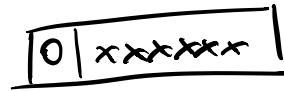
c = '\Uxxxx'

c = '\xxx'

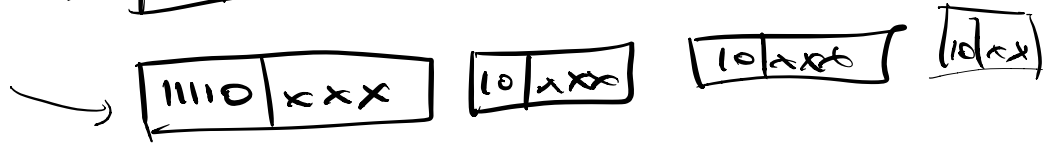
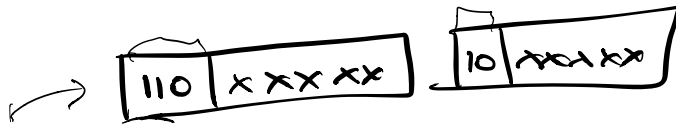
fmt.Println(string(c))

UTF-8

Us-ASCII (0 → 127)



> 127



string

= sequenza di byte
(solitamente usata per
rappresentare sequenze di
caratteri in UTF-8)

var s string

$S = \underbrace{\text{"Paolo"}}_{\text{lettere di tipo}} \text{ sfrigge interpretato.}$

$S = \text{"Nicolò"}$

S = "ciao" in natura

$\Sigma =$ "مذا \ in paper"

func. Print(s)

$\rightarrow S = \{ \underbrace{x_{11}, x_{12}, x_{14}}_{\text{row 1}}, x_{17} \}$

10100001

S= 'ciao

qui niente
viene interpretato

letterale di tipo stringa
raw

FUNZIONI E OPERATORI SU STRINGHE

1)

`len(s)`

→ restituisce la
lunghezza in
BYTE di s

```
var s string  
s = "citta"
```

```
fmt.Println(len(s))
```

2)

`s1+s2`

→ una nuova
stringa
ottenuta
concatenando
s1 e s2

```
var s1, s2 string
```

```
s1 = "sce"
```

```
s2 = "no"
```

```
s := s1 + s2
```

```
fmt.Println(s)
```

3)

`s[i]` → i-esimo byte
di `s`
(0, ..., `len(s)-1`)

```
for i:=0; i<len(s); i++  
    ... s[i] ...  
}
```

4)

`fmt.Scan(&s)`

↳ legge una stringa
whitespace-separated

5)

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

```
→ b := bufio.NewReader(os.Stdin)
```

```
for b.Scan() {
```

```
    s := b.Text()  
    ...  
}
```

legge la
prossima
riga

Ctrl-D

6)

$s[i:j]$ → produce una
sottstringa
dall'indice i
compreso all'indice
 j escluso

```
var s1, s2 string
s1 = "ma ma ma"
      0 1 2 3 4
s2 = s1[1:3]
s2 = s1[0:4]
s2 = s1[:3]
s2 = s1[3:]
s2 = s1[:]
```

indice del
byte a cui
il carattere
inizia / runa

7)

```
for i, x := range s {  
    ...  
}
```

↓
scandisce la stringa
s carattere per carattere

for indice, una := range stringa

ESERCIZI

- leggere una stringa e stampare quante 'a' contiene
 - caso ASCII
 - caso generale
- stessa cosa ripetuta decide (il programma ripetuta volte di inserire una riga e dopo ogni riga stampa quante 2 contiene).
- Alla fine stampa il numero medio di 2 per riga (float64)
- Data una stringa ASCII, stampare tutto ciò che compare prima della prima