```
// 英文占1字节
// 中文占3字节
func charDemo() {
   // step:1
   s := "Yes我爱Go" // utf-8可变长
   for _, v := range []byte(s) {
      fmt.Printf("%X ", v)
   }
   fmt.Println()


   // step:2
   for i, ch := range s { // ch is a rune (string -> utf-8解码 -> 每个字符 -> 转unicode -> 最后放到rune类型中)
      fmt.Printf("(%d %c)", i, ch)
   }
   fmt.Println()

   // step:3
   fmt.Println("Rune count:", utf8.RuneCountInString(s))
   bytes := []byte(s)
   for len(bytes) > 0 {
      ch, size := utf8.DecodeRune(bytes)
      fmt.Printf("ch = %c, size = %v", ch, size)
      fmt.Println()
      bytes = bytes[size:]
   }


   // step:4 转rune
   runes := []rune(s)
   for i, ch := range runes {
      fmt.Printf("(%d %c)", i, ch)
   }
   fmt.Println()
}

* 结果:

59 65 73 E6 88 91 E7 88 B1 47 6F
(0 Y)(1 e)(2 s)(3 我)(6 爱)(9 G)(10 o)
Rune count: 7
ch = Y, size = 1
ch = e, size = 1
ch = s, size = 1
ch = 我, size = 3
ch = 爱, size = 3
ch = G, size = 1
ch = o, size = 1
(0 Y)(1 e)(2 s)(3 我)(4 爱)(5 G)(6 o)
```

```
rune 相当于go的char:

* 使用range遍历pos, rune对
* 使用utf8.RuneCountInString获得字符串数量
* 使用len获得字节长度
* 使用[]byte获得字节
```

```
字符串其他操作:

* Fields, Split, Join 分割字符串
* Contains,  Index    查找子串
* ToLower,  ToUpper
* Trim, TrimRight, TrimLeft
```