Codificación de caracteres

Introducción a la codificación de caracteres

ascii

En los principios de la informática los ordenadores se diseñaron para utilizar sólo caracteres ingleses, por lo tanto se creó una codificación de caracteres, llamada ascii (American Standard Code for Information Interchange) que utiliza 7 bits para codificar los 128 caracteres necesarios en el alfabeto inglés. Posteriormente se extendió esta codificación para incluir caracteres no ingleses. Al utilizar 8 bits se pueden representar 256 caracteres. De esta forma para codificar el alfabeto latino aparece la codificación ISO-8859-1 o Latín 1.

Unicode

La codificación unicode nos permite representar todos los caracteres de todos los alfabetos del mundo, en realidad permite representar más de un millón de caracteres, ya que utiliza 32 bits para su representación, pero en la realidad sólo se definen unos 110.000 caracteres.

UTF-8

UTF-8 es un sistema de codificación de longitud variable para Unicode. Esto significa que los caracteres pueden utilizar diferente número de bytes.

La codificación de caracteres en python3

En Python 3.x las cadenas de caracteres pueden ser de tres tipos: Unicode, Byte y Bytearray.

- El tipo `unicode` permite caracteres de múltiples lenguajes y cada carácter en una cadena tendrá un valor inmutable.
- El tipo `byte` sólo permitirá caracteres ASCII y los caracteres son también inmutables.
- El tipo `bytearray` es como el tipo `byte` pero, en este caso, los caracteres de una cadena si son mutables.

Algo que debe entenderse (e insiste Mark Pilgrim en su libro *Dive into Python*) es que "los bytes no son caracteres, los bytes son bytes; un carácter es en realidad una abstracción; y una cadena de caracteres es una sucesión de abstracciones".

Funciones chr() y ord()

- `chr(i)` : Nos devuelve el carácter Unicode que representa el código `i` .

```
>>> chr(97)
'a'
>>> chr(1004)
'б'
```

- `ord(c)` : recibe un carácter `c` y devuelve el código unicode correspondiente.

```
>>> ord("a")
97
>>> ord("б")
1004
```