

# 81-CRIPT-codificacion

March 5, 2018

Limpiar un texto de caracteres indeseados

```
In [1]: alfb = "ABCDEFGHJKLMNOPQRSTUVWXYZ" #Caracteres permitidos
```

```
In [2]: texto = "Through the use of abstraction and logical reasoning, mathematics developed from counting and calculation."
```

La codificación ASCII representa cada caracter mediante un entero entre 0 y 255. Como  $256 = 2^8$ , cada uno de esos enteros se representa en binario mediante 8 bits (ceros o unos), que es igual a 1 byte. Entonces un texto de  $N$  caracteres ocupa en memoria  $N$  bytes. La función *ord* nos da el entero, en ASCII, que corresponde a un caracter ( $\text{ord}('A') = 65$ ), mientras que *chr* es su función inversa.

```
In [3]: print map(ord,[x for x in alfb])
```

```
[65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88]
```

```
In [4]: print map(chr,map(ord,[x for x in alfb])) #Comprobamos que chr es la inversa de ord
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
```

```
In [5]: from string import * ##Hace falta para que funcione join
```

```
def limpiar(texto,alfb):
```

```
    '''Dejamos en el texto solo los caracteres de alfb. '''
```

```
    L = [x.capitalize() for x in list(texto)] #Cambiamos el texto a mayusculas
```

```
    L1 = [car for car in L if car in alfb]
```

```
    C1 = join(L1,sep = "")
```

```
    return C1
```

```
In [6]: limpiar(texto,alfb)
```

```
Out[6]: 'THROUGHTHEUSEOFABSTRACTIONANDLOGICALREASONINGMATHEMATICSDEVELOPEDFROMCOUNTINGCALCULATION.'
```

Codificación más adaptada al tamaño del alfabeto

En ocasiones, en lugar de usar la codificación ASCII queremos codificar los caracteres, suponiendo que tenemos 26 como en *alfb*, mediante enteros del 0 al 25. Para eso debemos re-definir las funciones *ord* y *chr*:

```
In [7]: L_alfb = list(alfb)
```

```

In [8]: def ord2(c):
        return L_alfb.index(c)

In [9]: def chr2(n):
        return L_alfb[n]

In [10]: print map(ord2,[x for x in alfb])

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

In [11]: print map(chr2,map(ord2,[x for x in alfb]))

['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S']

```

Una codificación binaria

En este ejercicio, que se repite y se utiliza en la hoja "87-CRIPT-mhellman", codificamos cada letra mediante un entero entre 0 y 25, y luego cada entero como una palabra de 5 bits.

```

In [12]: alfb = "ABCDEFGHJKLMNOPQRSTUVWXYZ"

In [13]: L_alfb = list(alfb)

In [14]: texto = "Through the use of abstraction and logical reasoning, mathematics developed :

In [15]: def ord2(c):
        return L_alfb.index(c)

In [16]: def chr2(n):
        return L_alfb[n]

In [17]: print map(ord2,[x for x in alfb])

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

In [18]: (7).digits(base=2,padto=5)

Out[18]: [1, 1, 1, 0, 0]

```

*Digits* escribe los dígitos de izquierda a derecha. Como nos interesa el orden inverso en la función *cadena*, cuarta línea, invertimos el orden.

```

In [19]: def cadena(L):
        '''Convertimos una lista en cadena de caracteres'''
        C = ''
        L.reverse()
        for item in L:
            C = C+str(item)
        return C

```



```
In [30]: def descod(C):
```

```
    C1 = ''
```

```
    while C != '':
```

```
        C1 = C1+chr2(ZZ(C[:5],base=2))
```

```
        C = C[5:]
```

```
    return C1
```

```
In [31]: descod(texto_cod3)
```

```
Out[31]: 'THROUGHTHEUSEOFABSTRACTIONANDLOGICALREASONINGMATHEMATICSDEVELOPEDFROMCOUNTINGCALCULAT
```

```
In [32]: descod(cod(texto2)) #Parecen funciones inversas
```

```
Out[32]: 'THROUGHTHEUSEOFABSTRACTIONANDLOGICALREASONINGMATHEMATICSDEVELOPEDFROMCOUNTINGCALCULAT
```