83-CRIPT-vigenere-frec

March 4, 2018

```
In [1]: alfb = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

In [2]: L_alfb = list(alfb)

In [3]: texto = "Through the use of abstraction and logical reasoning, mathematics developed fi

In [4]: def ord2(c):
            return L_alfb.index(c)

In [5]: def chr2(n):
            return L_alfb[n]

In [6]: 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 [7]: 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'


In [8]: from string import *
        def limpiar(texto,alfb):
            L = map(ord,[x.capitalize() for x in list(texto)])
            L1 = [item for item in L if item in map(ord,[x for x in alfb])]
            C1 = join(map(chr,L1),sep = "")
            return C1

In [9]: texto_l = limpiar(texto,alfb);texto_l

Out[9]: 'THROUGHTHEUSEOFABSTRACTIONANDLOGICALREASONINGMATHEMATICSDEVELOPEDFROMCOUNTINGCALCULATI
```

Cifra de Vigenere

Puedes leer sobre este método de cifrado en Vigenere. Elegida una palabra clave, por ejemplo "CIRUELA", el método para encriptar consiste en, primero, obtener para cada letra de la clave el número ASCII que le corresponde:

```
In [10]: L3 = list("CIRUELA");print L3
```

```
['C', 'I', 'R', 'U', 'E', 'L', 'A']
```

```
In [11]: L4 = map(ord2,L3);print L4
```

```
[2, 8, 17, 20, 4, 11, 0]
```

Llamemos K a la lista de enteros correspondientes a la clave y m a la longitud de la clave, en nuestro ejemplo 7. Para cada resto módulo m, digamos k, tenemos un entero $K[k]$.

En el segundo paso del método de Vigenere, cada letra del mensaje, que ocupa una posición digamos N en el mensaje, se encripta de manera diferente según el valor del resto de dividir N entre m. Si el valor de ese resto es k, usamos k como clave para encriptarla mediante la cifra de César.

Finalmente, tenemos una lista de enteros, entre 0 y 25, y la transformamos en una cadena de caracteres, que son el mensaje encriptado.

```
In [12]: def encriptar_vig(T,C):
             '''T y C son cadenas de caracteres, T el texto y C la clave'''
             L1 = map(ord2,list(C))
             L2 = map(ord2,list(T))
             L3 = []
             for int in srange(len(L2)):
                 n = int%len(C)
                 L3.append(chr2((L2[int]+L1[n])%26))
             return join(L3,sep="")
```

```
In [13]: texto_e = encriptar_vig(texto_l,'CIRUELA');texto_e
```

```
Out[13]: 'VPIIYRHVPVOWPOHISMXCAEBZIRLNFTFAMNANZVUWZNKVXGEEHGURNMNSFMMYPZPGLWLSXCQCENMYGEICWYWA
```

```
In [14]: def complementaria(C):
             L1 = map(ord2,list(C))
             return join(map(chr2,[(26-n)%26 for n in L1]),sep="")
```

```
In [15]: encriptar_vig(encriptar_vig(texto_l,'CIRUELA'),complementaria('CIRUELA'))
```

```
Out[15]: 'THROUGHTHEUSEOFABSTRACTIONANDLOGICALREASONINGMATHEMATICSDEVELOPEDFROMCOUNTINGCALCULA
```

Análisis de frecuencias

```
In [16]: def cortar_texto(texto,nclave):
             '''Almacenamos en C una lista con nclave cadenas de caracteres, cada una
             contiene la parte del texto que se ha encriptado con la misma clave'''
             C=[item for item in texto[:nclave]]
             for i in srange(nclave):
                for j in srange(len(texto)):
                     if j%nclave==i:
                         C[i] += texto[j]
                     else:
                         continue
             return C
```

```
In [17]: cortado = cortar_texto(texto_e,7);cortado
```

```
Out[17]: ['VVVHEFNKGFGQEVUVUVAJFUKGECKGCKUMVTVWGUTGGOCWNOVGVKQPTCPCPQTYUHQNCTVQOFTQUTC',
          'PPPIBTZVUMLCIQCIGQWIUWKKBBKMVBNIMLZAVBMMUWJKMIQTIDEBMVUBVVIQKQDMXMPNIQGVBMG',
          'IIVSZFVXRMWECFIEJTWGFWRKZYJERPRJEJZRKRUBRJCCDKTFIVGZETRZFJTKZTVUZRVDKJKKFJ',
          'IIOMIAUGNYLNWHYXNMNYNJFMWYBUWZLQLYALMJCGNNSCYBMJYFUFUYNWPCNBYXLNXMLUCWBCNY',
          'YYWXRMWEMPSMYQQXIXLWMLSTEQELXSFVIBSKJTREMRMHRIHIPCGXMALEERMRRMMSMIEXGSERLR',
          'RRPCLNZENZXYWPPSXFPLZJMCWLDFTCLTNTCFTPRENZYDEXPOLDPSDSPWEEYPEDPLYTESLGEFPE',
          'HHOANANHSPCGAANEADSNNSJAMTBMVACTOSOMRARHSTEESAVATLUESEMIIEGWICSRCNEELECEPD']
```

```
In [18]: def analisis_frec(T):
             frecuencias = {}
             N = len(T)
             for letra in T:
                 if letra in frecuencias:
                     frecuencias[letra] += (1/N).n()
                 else:
                     frecuencias[letra]=(1/N).n()
             return frecuencias
```

```
In [19]: def invertir(dicc):
             dict_inv = {}
             for key in dicc:
                 dict_inv[dicc[key]] = ord2(key)
             return dict_inv
```

```
In [20]: def analisis_frec_compl(T):
             dicc = analisis_frec(T)
             dicc2 = invertir(dicc)
             L = dicc2.items()
             L.sort(reverse=True)
             return L
```

```
In [21]: FF = analisis_frec_compl(cortado[0]);FF
```

```
Out[21]: [(0.146666666666667, 21),
          (0.106666666666667, 6),
          (0.0933333333333333, 19),
          (0.0800000000000000, 16),
          (0.0666666666666667, 10),
          (0.0533333333333333, 5),
          (0.0400000000000000, 15),
          (0.0266666666666667, 22),
          (0.0133333333333333, 24)]
```

```
In [22]: chr2(21)
```

```
Out[22]: 'V'
```

```
In [23]: chr2(6)
```

```
Out[23]: 'G'
```

Aparece aquí un problema: las frecuencias en el texto original, cuando se divide en 7 trozos, no corresponden a las frecuencias naturales en los textos en inglés y, entonces, es la G la que encripta a la E y no la V. Probablemente, eso indica que disponemos de demasiado poco texto. Si desencriptamos suponiendo que la clave consiste en encriptar E como V tendremos, siempre que no aparezca el mismo problema con los otros trozos del texto, una séptima parte de las letras mal.

```
In [24]: def analisis_frec2(T):
             dicc = analisis_frec(T)
             dicc2 = invertir(dicc)
             L = dicc2.items()
             L.sort(reverse=True)
             return chr2(L[0][1]-4)
```

£De dónde viene el −4 de la última línea?

```
In [25]: def buscar_clave(texto_e,nclave):
             cortado = cortar_texto(texto_e,nclave)
             clave = ''
             for i in srange(nclave):
                 clave += analisis_frec2(cortado[i])
             return clave

In [26]: buscar_clave(texto_e,7)

Out[26]: 'RINJILA'
```

Probemos con un texto mucho mayor:

```
him when the In [27]: texto_largo="... days of his fighting were done; ...

In [28]: texto_largo_l = limpiar(texto_largo,alfb)

In [29]: texto_largo_e = encriptar_vig(texto_largo_l,'CIRUELA');

In [30]: buscar_clave(texto_largo_e,7)

Out[30]: 'CIRUELA'
```

Longitud de la clave

```
In [31]: def buscar_clave2(texto_e):
             for int in srange(3,30):
                 clave = buscar_clave(texto_largo_e,int)
                 print clave

In [32]: buscar_clave2(texto_largo_e)
```

```
EEA
EEAE
EEAEE
EEEAEA
CIRUELA
EAAEEEEE
EEEEEEARA
AEELEURAAE
EAEAAEAEEEE
EREEEEAEALAA
AERAEEAREEAEA
CIRUELACIRUELA
ARAEELAEAEEEEEE
LLEEELEEERAEEEEA
RAEREEEAAULAAEEEE
EHAEELIELLEUEEEARA
RAAEAUEEREEELEAIEEA
LEAEEUUAAEAAELEERAAH
CIRUELACIRUELACIRUELA
EAUAEEAALUUEAAILHLREAL
AELAAELEUAEUAEAEALRELEA
LREUREAERAAAEEEEEUAEALEA
LELLUAAAIEERARAHJEEEEAAAE
AEIULRAEEULERUALAEELRAEEEL
EHEAELAREEHAEEAARALAAIRERLA
CIRUELACIRJELACIRUELACIRUELA
REERLEARUAEEERLELALAUEAEALEEA
```

£Qué se ve de interesante aquí?