Chapitre_Textes

December 8, 2019

0.1 Exercice 165 du Défi Turing

In [1]:
```
from functools import reduce

        f = open('dico.txt')
        histo = [0] * 1000
        for mot in f:
            valeur = reduce(lambda a, b : a * b, map(lambda c : ord(c) - 96, mot.rstrip()))
            if 2000 <= valeur < 3000:
                histo[valeur - 2000] += 1
        print(max((occ, 2000 + i) for (i, occ) in enumerate(histo)))
        f.close()

        # In [13]: reduce(lambda a, b : a * b, map(lambda c : ord(c) - 96, "chaud".rstrip()))
        # Out[13]: 2016

        # ##Réponse
        # In [20]: (executing cell "" (line 1 of "DefiTuring165.py"))
        # (50, 2160)
        #
        # In [22]: 50 * 2160
        # Out[22]: 108000
```

```
(50, 2160)
```

0.2 Exercices du manuel NSI Ellipses

0.3 Exercice 5

In [45]:
```
def unicode(s):
            """Affiche les caractères d'une chaine, leur point de code et leurs octets codant
            et en binaire"""
            for caractere in s:
                octets = caractere.encode("utf-8")
                octets_hexa = list(map(hex, octets))
                octets_bin = list(map(bin, octets))
                print("Caractère : {} | Point de code  : {} | Octets (hexa) : {} | Octets (bi
```

```
def unicode2(s):
    print("Caractères : ")
    for c in s:
        print(c, end=",")
    print("\n\nPoints de code : ")
    for c in s:
        print(ord(c),end=",")
    octets = s.encode("utf-8")
    print("\n\nOctets codants en hexadécimal : ")
    for c in octets:
        print(hex(c), end=',')
    print("\n\nOctets codants en binaire : ")
    for c in octets:
        print(bin(c), end=',')

def unicode3(s):
    """Affiche les caractères d'une chaine, leur point de code et leurs octets codant
    et en binaire"""
    for caractere in s:
        octets = caractere.encode("utf-8")
        octets_hexa = [format(b,'x') for b in octets]
        octets_bin = [format(b,'08b') for b in octets]
        print("Caractère : {} | Point de code  : {} | Octets (hexa) : {} | Octets (bi
```

```
In [25]: unicode("lycée")

Caractère : l | Point de code  : 108 | Octets (hexa) : ['0x6c'] | Octets (binaire) : ['0b11011
Caractère : y | Point de code  : 121 | Octets (hexa) : ['0x79'] | Octets (binaire) : ['0b11110
Caractère : c | Point de code  : 99 | Octets (hexa) : ['0x63'] | Octets (binaire) : ['0b110001
Caractère : é | Point de code  : 233 | Octets (hexa) : ['0xc3', '0xa9'] | Octets (binaire) : [
Caractère : e | Point de code  : 101 | Octets (hexa) : ['0x65'] | Octets (binaire) : ['0b11001
```

```
In [39]: unicode2("lycée")

Caractères :
l,y,c,é,e,

Points de code :
108,121,99,233,101,

Octets codants en hexadécimal :
0x6c,0x79,0x63,0xc3,0xa9,0x65,
```

```
Octets codants en binaire :
0b1101100,0b1111001,0b1100011,0b11000011,0b10101001,0b1100101,

In [46]: unicode3("lycée")

Caractère : l | Point de code  : 108 | Octets (hexa) : ['6c'] | Octets (binaire) : ['01101100']
Caractère : y | Point de code  : 121 | Octets (hexa) : ['79'] | Octets (binaire) : ['01111001']
Caractère : c | Point de code  : 99 | Octets (hexa) : ['63'] | Octets (binaire) : ['01100011']
Caractère : é | Point de code  : 233 | Octets (hexa) : ['c3', 'a9'] | Octets (binaire) : ['110
Caractère : e | Point de code  : 101 | Octets (hexa) : ['65'] | Octets (binaire) : ['01100101']
```

0.3.1 Exercice 228 page 273

Encodage UTF-8 d'un caractère en hexadécimal, décimal et binaire

```
In [4]: etoile = chr(8902)

In [6]: etoile_octet = etoile.encode('utf8')

In [7]: etoile_octet

Out[7]: b'\xe2\x8b\x86'

In [19]: etoile_liste_octet = [bin(octet) for octet in etoile_octet]
         print(etoile_liste_octet)

['0b11100010', '0b10001011', '0b10000110']


In [21]: etoile_liste_decimal = [octet for octet in etoile_octet]
         print(etoile_liste_decimal)

[226, 139, 134]
```

0.3.2 Exercice 230 page 273

```
In [71]: def longueur(b):
             """Retourne le nombre de caractères encodé par une
             chaine d'octets en utf8"""
             k = 0
             long = 0
             while k < len(b):
                 #attention les représentations binaires des octets ne sont pas remplies par d
                 binaire = bin(b[k]).lstrip('0b').zfill(8)
                 #decimal = b[k]
                 long += 1
                 if binaire[0] == '0':
                     k += 1
```

```
                elif binaire[:3] == '110':
                    k += 2
                elif binaire[:4] == '1110':
                    k += 3
                else:
                    k += 4
            return long
```

```
In [48]: longueur(etoile.encode('utf8'))
```

```
3
11100010
trois


Out[48]: 1
```

```
In [73]: chaine = 'élémentaire mon cher Watson'.encode('utf8')

In [75]: longueur('élémentaire mon cher Watson'.encode('utf8'))

Out[75]: 27

In [76]: len('élémentaire mon cher Watson')

Out[76]: 27

In [51]: chaine = 'élémentaire mon cher Watson'.encode('utf8')

In [52]: type(chaine)

Out[52]: bytes

In [54]: len(chaine)

Out[54]: 29

In [56]: chaine

Out[56]: b'\xc3\xa9l\xc3\xa9mentaire mon cher Watson'

In [66]: '1110'.zfill(8)

Out[66]: '00001110'
```