Série TP n°3

Chapitre 1: Codage

Module	**TAL** trait. Auto. Lang. natur.	
Filière	**Master ISIL**	**1ère Année**

Le codage

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# afficher tous les codages qui existes
import encodings
print ''.join('- ' + e + '\n' for e in sorted(set(encodings.aliases.aliases.values())))

# fixer le codage des caractères dans l'entête de script
# -*- coding: utf-8 -*-

# en python 2.7,
# il existe deux types de chaine de caractère le type str, et le type unicode

#python2.7
print type('chaine') # bits => encodée
print type(u'chaine') # unicode => décodée
print type(u'étudiant') # unicode => décodée
print type(u'السلام عليكم') # unicode => décodée

# python 3
type("chaine") # unicode => decodée
type(b"chaine") # bits => encodée

# pour lire le contenu d'un fichier utiliser le décodage
f = open("text.txt")
line = f.readline().decode('utf8')
# pour afficher utiliser  encoder
print line.encode('utf8')
print u"السلام عليكم".encode('utf8')
```

Le soundex

Soundex est un algorithme phonétique d'indexation de noms par leur prononciation en anglais britannique. L'objectif basique est que les noms ayant la même prononciation soient codés avec la même chaîne de manière à pouvoir trouver une correspondance entre eux malgré des différences mineures d'écriture. Soundex est le plus largement connu des algorithmes phonétiques et est souvent utilisé incorrectement comme synonyme de « algorithme phonétique ».

Historique

Soundex a été conçu par Robert Russell et Margaret King Odell et breveté en 1918 et 1922 (US patent 1,261,167 et 1,435,663). Une variante nommée *American Soundex* a été utilisée dans les années 1930 pour une analyse rétrospective des recensements américains entre 1890 et 1920.

Le code soundex s'est fait connaître dans les années 1960 lorsqu'il est devenu le sujet de nombreux articles dans les communiqués et le journal de l'Association for Computing Machinery, et tout spécialement décrit par Donald Knuth dans son magnum opus, *The Art of Computer Programming*.

Description

Le code soundex consiste pour chaque nom en une association d'une lettre suivie de trois chiffres : la lettre correspond à la re du nom, et les chiffres encodent les consonnes restantes. Les consonnes à prononciation similaire ont le même code, donc, par exemple, les lettres B, F, P et V sont toutes codées « 1 ». Les voyelles peuvent influencer le code d'une consonne, mais ne sont jamais codées directement (sauf bien sûr si c'est la première lettre du nom).

L'algorithme exact procède comme suit :

1. Supprimer les éventuelles 'espaces' initiales
2. Mettre le mot en majuscule
3. Garder la première lettre
4. Conserver la première lettre de la chaîne
5. Supprimer toutes les occurrences des lettres : a, e, h, i, o, u, w, y (à moins que ce ne soit la première lettre du nom)
6. Attribuer une valeur numérique aux lettres restantes de la manière suivante :
    - **Version pour l'anglais :**
        - 1 = B, F, P, V
        - 2 = C, G, J, K, Q, S, X, Z
        - 3 = D, T
        - 4 = L
        - 5 = M, N
        - 6 = R
    - **Version pour le français :**
        - 1 = B, P
        - 2 = C, K, Q
        - 3 = D, T
        - 4 = L
        - 5 = M, N
        - 6 = R
        - 7 = G, J
        - 8 = X, Z, S
        - 9 = F, V
7. Si deux lettres (ou plus) avec le même nombre sont adjacentes dans le nom d'origine, ou s'il n'y a qu'un h ou un w entre elles, alors on ne retient que la première de ces lettres.
8. Renvoyer les quatre premiers octets complétés par des zéros.

En effectuant cet algorithme, on obtient avec "Robert" et "Rupert" la même chaîne : "R163", tandis que "Rubin" donne "R150".

```python
# soundex.py
# Make dictionary numerics to map each letter to its group
groups = ['aehiouwy', #0
    'bfpv', #1
    'cgjkqsxz', #2
    'dt', #3
    'l', #4
    'mn', #5
    'r'] #6
numerics = {'a': '0',  'c': '2', 'b': '1', 'e': '0', 'd': '3', 'g': '2', 'f':
'1',
 'i': '0', 'h': '0', 'k': '2', 'j': '2', 'm': '5', 'l': '4', 'o': '0',
 'n': '5', 'q': '2', 'p': '1', 's': '2', 'r': '6', 'u': '0', 't': '3',
 'w': '0', 'v': '1', 'y': '0', 'x': '2', 'z': '2'}


def soundex(name):
    """ soundex module conforming to Knuth's algorithm
        implementation 2000-12-24 by Gregory Jorgensen
        public domain
    """
    # digits holds the soundex values for the alphabet
    sndx = ''
    firstchar = name[0].upper()
    name = name[0:] # le reste du mot

    # translate alpha chars in name to soundex digits
    for c in name.lower():
        d = numerics.get(c, '0')
        # duplicate consecutive soundex digits are skipped
        if not sndx or (d != sndx[-1]):
            sndx += d

    # replace first digit with first alpha character
    sndx = firstchar + sndx

    # remove all 0s from the soundex code
    sndx = sndx.replace('0','')

    # return soundex code padded to len characters
    sndx =  sndx + '0000'
    return sndx[:4]
```

```
if __name__ == "__main__" :
    words =[ "mohammed", "mohamad", "mohamd",
    "physique", "physik", "phosphore", "fosfor",
    ]
    for word in words :
        code = soundex(word.lower())
        print "%4s %-4s" % (code, word)
```

Travail à domicile :

1- Ecrire un programme qui permet la conversion d'un texte en langue Tamazight, entre les scripts Latin, Arabe, et Tifinagh.

2- On donne une dictionnaire de mots en français, représenté comme un fichier texte, où chaque mot est sur une ligne.

On veut rechercher un mot dans le dictionnaire d'une façon flou, par exemple, si l'utilisateur donne le mot « disk », on lui donne tous les mots comme « disque », disquettes, on peut suggérer à l'utilisateur des mots plus proches.