1 TD 9 : Expression régulières et discours des présidents

(correction page ??)

Abordé lors de cette séance	
programmation	expression régulières
algorithme	comptage

Dès qu'on cherche à analyser du texte de façon intelligente, on utilise nécessairement les expressions régulières qui offrent un moyen simple et rapide de chercher des motifs dans un texte. Tous les langages proposent maintenant les mêmes outils, le langage *Python* propose quant à lui le module `re` [1]. Il s'agit d'explorer cette fonctionnalité lors de cette séance.

Première demi-heure : données .

1) On récupère les voeux des présidents au soir du 31 décembre de quelques années et quelques présidents différents grâce au programme suivant. [2]

```
#coding:latin-1
import urllib, os, os.path
def charge_discours () :
    discours = { }
    for annee in [2001, 2005, 2006, 2007, 2008, 2009, 1974, 1975,
                    1979, 1983, 1987, 1989, 1990, 1994] :
        nom = "VOEUX%02d.txt" % (annee % 100)
        if os.path.exists (nom) :
            # si le fichier existe (il a déjà été téléchargé une fois)
            f = open (nom, "r")
            text = f.read ()
            f.close ()
        else :
            # si le fichier n'existe pas
            link = "http://www.xavierdupre.fr/enseignement/td_python/" + \
                     "python_td_minute/data/voeux_presidents/" + nom
            url = urllib.urlopen (link)
            text = url.read ()
            # on enregistre les données pour éviter de les télécharger une seconde fois
            f = open (nom, "w")
            f.write (text)
            f.close ()

        discours [annee] = text
    return discours

if __name__ == "__main__" :
    discours = charge_discours ()
    print "nombre de discours ", len(discours)
```

1. http://docs.python.org/library/re.html
2. ttp://www.xavierdupre.fr/enseignement/td_python/python_td_minute/data/voeux_presidents/VOEUX*.txt, ces données proviennent du blog d'Arthur Charpentier http://freakonometrics.blog.free.fr/index.php?post/2011/05/12/De-quoi-parle-un-president-francais-le-31-decembre.

On pourra enregistrer ce programme sous le nom `discours.py` et s'y référer en utilisant l'instruction :

```
import discours
textes = discours.charge_discours()
```

2) On va dans un premier temps construire des statistiques sur l'ensemble des textes plutôt que sur chaque texte pris séparément. Ecrire une fonction qui fait la concaténation de tous les textes.

```
import discours
textes = discours.charge_discours()

def somme_texte (textes):
    ...
    return ...
```

3) Les accents sont toujours un problème car une lettre majuscule n'a pas d'accent : l'ordinateur considère que "A", "à", "a" sont des mots différents. On écrit alors une fonction qui remplace les accents par des lettres non accentuées. La fonction fait aussi autre chose, qu'est-ce ?

```
accent = {     'à':'a', 'â':'a',  'ä':'a',
                    'é':'e', 'è':'e', 'ê':'e', 'ë':'e',
                    'î':'i', 'ï':'i',
                    'ù':'u', 'û':'u', 'ü':'u',
                    'ô':'o', 'ö':'o',
               }

def pas_daccent (texte) :
    res = ""
    for c in texte :
        c = c.lower ()
        res += accent.get (c,c)
    return res
```

Seconde demi-heure : compter les mots .

4) On écrit une fonction à compléter qui découpe en mots avec la fonction `split` :

```
def decoupe_mot (texte) :
    return ....
```

5) On écrit une fonction à compléter qui compte la fréquence de chaque mot distinct :

```
def compte_mots (mots) :
    d = { }
    for mot in mots :
        ...
        ...
    return d
```

6) On écrit une fonction à compléter qui trie les mots par ordre de fréquence décroissant :

```
def mot_tries (d):
    l = [ (n,m) for ... in d.iteritems () ]
    l.sort (reverse = True)
    return l
```

7) On veut afficher les 25 mots les plus fréquents, comment utiliser les trois fonctions précédentes ? Que pensez-vous de ces mots ?

Troisième demi-heure : expression régulières

8) En vous aidant de la page http://docs.python.org/library/re.html, expliquez ce que fait la fonction suivante ?

```
def decoupe_mot2 (texte) :
    exp = re.compile ("\\w+", re.IGNORECASE)
    return exp.findall(texte)
```

9) Comparer les résultats avec la dernière question de la partie précédente. Pourquoi le mot `l` est-il présent ?

10) Comment modifier l'expression "\\w+" pour ne compter que les mots de plus de six lettres ?

Quatrième demi-heure : expressions plus complexes

11) Toujours en vous aidant de la page http://docs.python.org/library/re.html, expliquez ce que fait la fonction suivante ?

```
def trouver_expression(texte) :
    exp = re.compile ("je .{1,60}", re.IGNORECASE)
    return exp.findall(texte)
```

12) Utiliser cette fonction sur le texte entier puis sur chaque texte pris séparément.

Pour aller plus loin ou pour ceux qui ont fini plus tôt

13) Avec la même expression régulière, rechercher indifféremment le mot *securite* ou *insecurite*.