TP1

Le TP sera réalisé en python 3.

La librairie `re` de python permet d'utiliser les expressions régulières pour effectuer de la recherche dans les textes.

```
import re
```

1. Expressions régulières

Exercice 1.1 : recherche de motifs

La recherche dans un texte nécessite de définir un motif. Ce motif est défini dans une chaîne brute (qui n'interprète pas les caractères spéciaux). Une chaîne brute est préfixée par un `r`. exemple : `r'bonjour\n'` est la chaîne contenant les 9 caractères b,o,n,j,o,u,r,\,n

Pour rechercher un motif `m` dans une chaîne `s`, nous utiliserons la commande `re.findall()` qui retourne la liste des mots correspondant au motif

Nous cherchons dans le texte `"Un éléphant, ça trompe énormément!"` tous les mots terminant par `"nt"`.

Testez l'exemple suivant :

```
texte = "Un éléphant, ça trompe énormément!"
liste_mots = re.findall(r'\w+nt',texte)
for i in range(len(liste_mots)) :
  print liste_mots[i]
```

Puis testez l'expression sur le texte `"L'éléphant entre dans l'entrepôt de porcelaine."` Des portions de mots `ent` sont reconnues, or seul le mot éléphant devrait être reconnu dans ce cas.

Les "ancres" permettent de tester la position d'un motif :

- ^: début de ligne
- $: fin de ligne
- \b : début ou fin de mot

Modifiez l'expression régulière afin de reconnaître uniquement (sur cet exemple) le mot `"éléphant"`.

Exercice 1.2

Voici un petit texte qu'on pourra améliorer à son goût :

```
"Il y a 2 mois, ce n'est pas toi qui a découvert cette vieille armoire,
cachée sous la toiture.\n
```

```
Moi, je te dis que c'est bien moi, il y a 2 ou 3 mois."
```

On demande d'écrire les motifs, puis de tester, pour chercher :

1. L'un des mots "moi" ou "toi", partout dans le texte;
2. tous les mots contenant "moi" ou "toi".

Exercice 1.3 : Extraction

Les parenthèses servent à mémoriser un élément. Testez l'exemple suivant qui extrait l'année, le mois et le jour de mois:

```
import re
une_date = "2002-12-16"
ma_date = re.findall(r'(\d+)-(\d+)-(\d+)',une_date)
print('jour :', ma_date[0][2])
print('mois :', ma_date[0][1])
print('année :', ma_date[0][0])
```

Nous souhaitons modifier le format de date retournée par la commande système date. (il faut pour cela faire appel à la librairie os : `import os`)

```
s = os.popen("date").readline();
```

par exemple `jeudi 22 mars 2012, 09:02:19 (UTC+0100)`

En extraire les différents éléments pour afficher :

```
Nous sommes jeudi, 22ème jour du mois de mars de l'année 2012. Voici
maintenant l'heure : il est 09 heures 02 minutes et 19 secondes.
```

Exercice 1.4 : Reconnaître un nombre décimal

Il s'agit d'extraire d'un texte tous les nombres décimaux correctement écrits qu'il contient (signe compris).

```
Depuis vendredi, les températures sur le nord du pays ont atteint des
valeurs remarquables pour la saison (+8 à +11 °C par rapport aux normales),
le seuil de chaleur (25 °C) étant souvent atteint. À Paris-Montsouris, ce
seuil a été atteint 4 jours consécutivement, depuis vendredi 19 avril, dont
une température maximale de 27,2 °C le samedi 20. À la station du Cap-de-la-
Hève près du Havre, il a déjà fait plus de 20 °C cinq jours consécutivement,
ce qui reste très rare à cette période de l'année même si l'on trouve une
série comparable l'année passée, exactement aux mêmes dates, au sein d'une
période de chaleur précoce exceptionnelle à l'échelle de la métropole.
D'autres séries de températures remarquables pour des stations historiques
ont été établies de la Normandie au nord de la Seine, mais là encore celles
de l'année passée sont le plus souvent d'intensité légèrement supérieure
```

```
pour une durée similaire. Deux autres séquences similaires ont pu être
observées dans un passé récent (23 au 29 avril 2007 et 18 au 23 avril 2011)
mais il y a très peu d'équivalent auparavant (on peut citer pêle-mêle 1947,
1949, 1984, 1993).
Au cours du week-end prolongé, il a ainsi fait plus chaud sur une moitié
nord qu'au sud. Plus précisément, la valeur de l'indicateur de température
maximale pour la région Nord, calculé à partir des stations de référence
entre vendredi 19 et mardi 22, est la plus élevée (24,8 °C, contre 23,2 °C
pour la région Sud-Est et 21,2 pour la région Sud-Ouest). Les températures
maximales ont toutefois dépassé les 26 °C sur le Sud-Est lundi 22 (jusqu'à
27 °C à Avignon ou Nîmes).
```

(source : Météo France)

Pour chaque nombre extrait, on demande d'afficher séparément le signe ainsi que les parties entière et décimale.

Exercice 1.5 : Chercher des mots dans un texte

Soit un texte stocké dans le fichier texte (par exemple : declaration.txt). Pour ouvrir le fichier dont le nom est fourni en argument, on utilisera la commande :

```
f = open('mon_fichier.txt','r')
```

On demande à l'utilisateur de saisir le mot recherché. Le script doit parcourir chaque ligne du fichier et afficher chaque ligne où le mot est présent en le mettant en valeur en l'entourant par exemple de « ». Conclure l'étude par une phrase du genre :

```
le mot .... est présent ... fois dans .. lignes du fichier ...
le mot .... n'a pas été trouvé dans le fichier ...
```

Exercice 1.6 : Traitement d'une archive de messages

Il s'agit d'extraire d'un fichier de messagerie les éléments principaux : la date, l'expéditeur, le destinataire, le sujet et le texte principal (liste de messages à trouver dans `~/Maildir`)

Voir : Maildir

1. ouvrir un de vos fichier mails et parcourir ses lignes
2. écrire les motifs nécessaires pour extraire l'expéditeur, le destinataire et le sujet, précédés du numéro de ligne

Attention ! le sujet peut comporter `Re:` comme dans `Subject: Re: PHP et les directory`. Dans ce cas l'éliminer.

Voici un exemple : un extrait d'un message et l'affichage souhaité

Message d'origine

```
Received: by mail.egim-mrs.fr (Postfix, from userid 331) id
4143F220C4; Mon,  2 Nov 2004 10:30:39 +0100 (CET)
Date: Mon, 2 Nov 2004 10:30:39 +0100
From: XXX YYY <xxx.yyy@ec-mrs.fr>
To: linux@nnx.com
Subject: Re: PHP et les directory
Message-Id: <20021202093039.GC27512@egim-mrs.fr>
References: <3DEB246D.8529C378@a4-interactive.com>
MIME-Version: 1.0
Content-Type: text/plain;
charset=iso-8859-1
Content-Disposition: inline
Content-Transfer-Encoding: 8bit
In-Reply-To: <3DEB246D.8529C378@a4-interactive.com>
User-Agent: Mutt/1.4i
```

Résultat du script :

```
30 Expéditeur : XXX YYY <xxx.yyy@ec-mrs.fr>
31 Destinataire : linux@nnx.com
32 Sujet : PHP et les directory
```

Exercice 1.7 : Reconnaissance des hyperliens d'une page WEB

Il s'agit d'obtenir la liste des URL incluses dans tous les hyperliens de la page, chacun étant accompagné du texte associé. On pourra tester sur la page extrait.html fournie.

- récupérer le nom du fichier,
- l'ouvrir en lecture,
- appliquer le motif sur chaque ligne
- construire un dictionnaire contenant le couple (url, libellé)
- afficher le résultat de l'analyse comme ci-dessous :

Résultat du script :

```
L'environnement graphique KDE  ---> kde-linux.html
système de fichiers ---> systemes-fichiers.html
Démarrer (sous) Linux ---> demarrage-linux.html
Exemple d'installation ---> install-mandrake-cfipen.html
Installation d'applications et archivage   ---> install-
archivage.html
```

```
Le service d'impression ---> impression-linux.html
TP Extension du système ---> tp-extension-systeme.html
commandes utilisateurs ---> commandes-generales.html
installation par NFS ---> install-reseau.html
serveur X  ---> x11r6-linux.html
```

2 - Arbre de complétion

Un algorithme de complétion est un mécanisme logique permettant d'anticiper la saisie et de proposer des mots automatiquement pour faciliter les recherches dans un formulaire sur une page web par exemple.

On utilise pour cela une structure de données arborescente, où chaque nœud de l'arbre est une lettre, ses nœuds enfants les lettres suivantes possibles du mot, avec un indicateur par lettre pour savoir si celle-ci est finale ou non.

Le but de cet exercice est de construire un arbre de complétion à partir de mots de vocabulaire, puis de l'utiliser pour compléter un début de mot proposé par l'utilisateur.

2.1 tests

Un arbre de complétion sera défini de manière récursive. Un nœud de l'arbre contient 2 éléments :

- fils : un dictionnaire de nœuds indexés par des caractères
- compteur : un entier valant :
  - 0 si le mot n'est pas dans la base
  - une valeur >0 indiquant le nombre d'occurrences du mot dans la base sinon.

Vous devez :

- Créer une classe ArbreComplétion contenant les attributs mentionnés.
- Définir un constructeur qui crée un nœud vide.
- Définir la méthode `insere(mot)` qui insère un mot dans un nœud. Cette méthode récursive teste si la première lettre du mot est présente dans le dictionnaire. Si non, elle crée l'entrée correspondante. Elle vérifie ensuite si le mot est un caractère isolé. Si oui, elle incrémente le compteur du nœud fils, sinon, elle insère la suite du mot dans le nœud fils.
- Définir une méthode `affiche(noeud)` qui affiche l'arbre de complétion de manière récursive.

Créez un arbre de complétion. Insérez les mots `'bonjour'`, `'bonjour'`, `'bonsoir'`, `'bon'`, `'jour'`, `'soir'` et vérifiez grâce à l'affichage que les mots sont correctement insérés.

Écrivez ensuite une fonction `suivant(debut, A)` qui, à partir de la chaîne `debut` et de l'arbre de complétion `A`, retourne la liste de mots qui complète le début de mot.

Améliorez votre fonction pour que les mots les plus courants apparaissent en premier dans la liste.

2.2 Vocabulaire complet

Une fois que tout fonctionne, vous devez créer un arbre contenant l'intégralité du vocabulaire français. Pour cela, vous devez télécharger le fichier : Lexique382.zip

- ouvrez le fichier
- créez une liste de vocabulaire vide
- pour chaque ligne du fichier, ajouter à la liste de vocabulaire le premier mot de la ligne
- ajouter chaque mot de la liste de vocabulaire dans l'arbre de complétion
- vérifiez que l'arbre reconnaît correctement les mots de vocabulaire rentrés par l'utilisateur