TP 2
Hiérarchies de données en XML

Le TP sera réalisé en python.
N'oubliez pas de démarrer vos scripts par les deux lignes :

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

Les fichiers xml permettent de stocker ou transmettre des données de type texte organisées de manière hiérarchique, à la façon d'un document contenant des chapitres, sous-chapitres, sections... Le document est structuré à l'aide de balises ouvrantes et fermantes : `<balise> … </balise>`, qui encadrent une portion de texte :
Exemple : `<titre> Bilan d'activité 2009-2010 </titre>`.
Chaque portion de texte peut-elle-même contenir des portions de textes encadrées par des balises et ainsi de suite... donnant au document sa structure hiérarchique...
Exemple :

```
<chiffre_d_affaires> 12456
      <premier_semestre> 7866 </premier_semestre>
      <second_semestre> 4590 </second_semestre>
</chiffre_d_affaires>
```

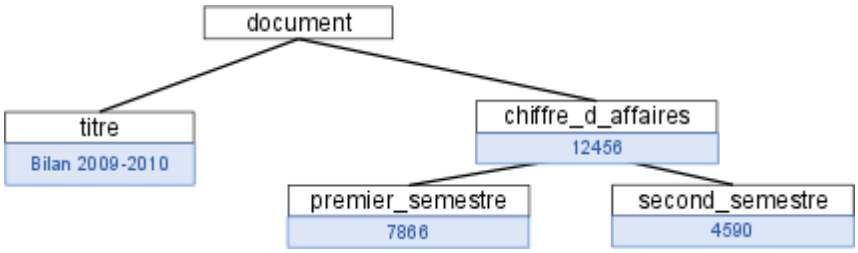
Le traitement des fichiers xml est facilité en Python à l'aide de la librairie etree :

```
import xml.etree.cElementTree as etree
```

qui construit une sructure de données arborescente à partir d'un fichier xml:

```
f = open ('bilan.xml')
doc = etree.parse(f)
```

l'objet doc (représentant le document) contient la structure de données :




```
r = doc.getroot()
```

la variable `r` représente la racine de l'arbre.
`r.tag` donne le nom de la balise
`r.text` donne la portion texte correspondante (éventuellement nulle)
`r.attrib` est un dictionnaire contenant des valeurs d'attributs de la balise
Enfin, `r` est un objet de type liste. Ainsi : `r[0]` représente le premier fils, `r[1]` le second fils etc...
la boucle :

```
for s in r :
    ...
```

permet de parcourir la liste des fils de `r`

Exo 1 :
On souhaite réaliser un convertisseur de monnaies à partir d'un fichier contenant des taux de conversion actualisés de l'euro vers d'autres monnaies (dollar, yen, etc...).
Le fichier http://www.ecb.int/stats/eurofxref/eurofxref-daily.xml contient les taux de conversions

mis à jour régulièrement, sous la forme d'un fichier au format xml.
pour ouvrir ce fichier, situé sur le web, on utilisera la librairie urllib :

```
from urllib import *
f = urlopen("http://www.ecb.int/stats/eurofxref/eurofxref-daily.xml")
```

Affichez le contenu de ce document.
Vous remarquerez qu'il est organisé en 3 sections : subject, sender, cube

- 1 - Ecrivez une fonction qui lit le premier niveau, c'est à dire qui affiche le nom de balise, le texte et les attributs des 3 sections principales du document.

- 2 - Selon la norme adoptée ici, les balises de type cube servent à stocker les données sous forme de listes : `<cube attrib1 = valeur1 attrib2 = valeur2....> </cube>`
Le premier sous-niveau représente la date (`time`),

```
<cube time="2011-09-23"> … </cube>
```

et le second sous-niveau contient les valeurs de taux de change sous forme d'attributs `currency` (devise) et `rate` (taux de change).

```
<cube currency="USD" rate="1.3430"/>
```

Ainsi, pour obtenir les taux de change, il extraire les attributs de chaque fils du premier sous-niveau : si `a` est un fils du premier sous-niveau, alors `a.attrib['currency']` contient le nom de la devise et `a.attrib['rate']` contient le taux de change.

Ecrivez une fonction `taux_de_change` qui reçoit le nom d'une devise et retourne le taux de change.

- 3 - Ecrire une fonction `convertisseur` qui reçoit un montant en euro et un nom de devise et retourne un montant dans cette devise.

Toutes ces fonctions seront testées au niveau du programme principal

Exo 2 : parcours récursif

On considère le fichier `clients.xml` contenant une liste de clients, décrits par leurs nom, prénom et adresse.
Nous utiliserons comme dans l'exercice précédent la librairie `etree` pour charger le contenu du fichier.

- 1 - Ecrire une fonction `affiche` qui affiche le contenu d'un arbre sous la forme d'une suite de lignes nom_de_balise : texte
c'est à dire :

```
répertoire :
client :
nom : Duval
prenom : Bertrand
adresse :
voie : 18, rue des Lilas
code_postal : 31000
```

etc...

remarque : cette fonction doit être *récursive* pour afficher l'arbre en "profondeur d'abord"

Modifier la fonction pour mettre en évidence visuellement (avec des tabulations par exemple) la profondeur des différents noeuds.

```
répertoire :
    client :
        nom : Duval
        prenom : Bertrand
        adresse :
            voie : 18, rue des Lilas
            code_postal : 31000
```

etc...

- 2 - Ecrire une fonction **`cherche_adresse`** qui reçoit un nom de client et affiche son adresse.