TP 1
Données structurées : Manipulation de fichiers de données

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

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

Exo 1 (rappel python) : fonctions, listes

1. Une année bissextile est multiple de 4 mais pas multiple de 1000. Ecrire une fonction qui prend en argument une année (entier) et retourne un booléen indiquant si l'année est bissextile.

2. On considère la liste donnant le nombre de jours des mois de l'année :

```
jours_mois = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

écrire une fonction qui reçoit une date (jour, mois, année) et retourne le nombre de jours depuis le premier janvier. Attention, si l'année est bissextile, il faut rajouter un jour au mois de fevrier.

Toutes ces fonctions seront testées par un programme principal.

Exo 2 : lecture de fichier texte et traitement des données

On considère une série d'enregistrements concernant des ventes réalisées par un exportateur de véhicules miniatures. Pour chaque vente, il entre dans son registre de nombreuses informations :
- nom de la société cliente
- nom et prénom du contact, adresse, téléphone
- nombre d'unités vendues
- prix de vente
- etc...

Ces informations sont stockées dans un fichier au format 'csv' (comma separated values) :
`ventes.csv`
Récupérez ce fichier sur Claroline.

Dans un premier temps, regardez son contenu avec un editeur de texte (geany, gedit ou autre...). La première ligne contient les noms des attributs (NUM_COMMANDE, QUANTITE,...). Les ligne suivantes contiennent les valeurs d'attributs correspondant à une vente donnée.
en tout plus de 2000 ventes sont répertoriées dans ce fichier.

Ouvrez-le maintenant à l'aide d'un tableur (gnumeric ou oocalc).
Les données sont maintenant "rangées" en lignes et colonnes pour faciliter la lisibilité.

L'exercice sera codé en Python. Vous pouvez effectuer les parties A et B en mode "ligne de commande" (sans oublier de garder une trace des commandes que vous avez tapées). Pour la

partie C, vous devez utiliser un éditeur de textes (geany par exemple) et exécuter le script dans un terminal.

A - Lecture des données en Python

Ouverture du fichier en python :

Créez maintenant un script Python.
Ouvrez ce fichier avec la commande:

```
f = open('ventes.csv', 'r')
```

Stockez ensuite la totalité du fichier dans une liste `L`.

rappel : on peut lire ensuite le fichier soit :

- dans sa totalité avec `f.read()` : le résultat est une chaine de caractère contenant tout le fichier,
- dans sa totalité en séparant les lignes avec `f.readlines()` : le résultat est une liste dont chaque élément est une ligne du fichier,
- ligne par ligne avec `f.readline()` : Le résultat est une chaîne de caractère. A chaque appel, une nouvelle ligne est lue.

Une fois nos manipulations sur le fichier effectué, on ferme le fichier avec la commande `f.close()`.

Définissez une variable `n` contenant le nombre d'éléments de cette liste.

Récupération de la liste des attributs :

La première ligne `L[0]` est une chaîne de caractères contenant la liste des attributs. On remarque que les différents attributs sont séparés par des virgules (la virgule est donc le *caractère de séparation*.)
Définissez une liste d'attributs `attr` à partir de la chaîne `L[0]` en utilisant la commande `split`. (la commande `s.split(c)` permet de construire une liste à partir d'une chaîne de caractères `s` et un caractère séparateur `c`)
Définissez la variable `m` contenant le nombre d'attributs .

Extraction d'une ligne :

Les autres lignes (L[i], i > 0) contiennent des valeurs d'attributs.
Affichez la chaîne de caractères `L[1]`

Chaque ligne doit être "découpée" pour extraire les différentes valeurs.
Prenez la ligne d'index 1 et extrayez les différents éléments dans une liste `val`.
Affichez la liste `val`. Vérifiez que `val` contient bien `m` éléments.

Construisez un dictionnaire `d` tel que pour tout $j \in 0,...,m-1$: `d[attr[j]] = val[j]`
Affichez ce dictionnaire.

Effectuez ces mêmes opérations avec la ligne d'indice 7. Il y a un problème!! lequel?

B - Librairie csv :

Pour lire "proprement" le contenu d'un fichier csv, on utilise la librairie `csv` :

```
import csv
```

Pour ouvrir `mon_fichier.csv` :

```
Lr = csv.reader(open("mon_fichier.csv","r"))
```

L'objet Lr contient une liste de lignes où chaque ligne correspond à un enregistrement, sous la forme d'une liste de valeurs.
Pour afficher le contenu du fichier, il y a donc besoin d'une double boucle :

```
for e in Lr:
      for val in e:
            print val
```

Remarque : l'objet Lr se comporte comme un flux (on ne peut pas accéder directement au ième élément `Lr[i]`).
Pour lire les enregistrements un par un, on utilise `Lr.next()`

Reprenez les opérations vues dans la partie précédente :

- création de la liste des attributs `attr` puis
- construction d'un dictionnaire pour la ligne d'index 1 (puis pour la ligne d'index 7)

quelle(s) différence(s) avec le cas précédent?

C - Tableau de dictionnaires

Pour effectuer plus commodément des opérations sur les données, on veut construire une *liste de dictionnaires*, nommée `D`, contenant la totalité des enregistrements (chaque élément `D[i]` est donc un dictionnaire qui contient les valeurs du ième enregistrement sous forme de couple (attribut : valeur))

Ecrire une fonction qui prend en argument le descripteur `Lr` et retourne `D`

Statistiques basiques

Le but est maintenant d'effectuer des statistiques simples : on souhaite connaître le nombre de ventes réalisées par pays.

Ecrire une fonction qui retourne le nombre de ventes par pays. Cette fonction prend en argument la liste D et retourne un dictionnaire `nb_ventes` contenant le nombre de ventes par pays.

Les clés de ce dictionnaire sont donc les noms de pays (Le nom de pays se trouve à `D[i]['PAYS']`)
Pour construire ce dictionnaire, on crée tout d'abord un dictionnaire vide `nb_ventes = {}` puis on parcourt les éléments de D :
- si le nom de pays `nom_pays = D[i]['PAYS']` n'est pas une clé de `nb_ventes`, on initialise `nb_ventes[nom_pays]` à 1.
- Dans le cas contraire, on incrémente `nb_ventes[nom_pays]`

remarque : le test `cle in nb_ventes` (ou `nb_ventes.has_key(cle)`) permet de verifier si la valeur `cle` est une clé du dictionnaire `nb_ventes`.

Testez cette fonction dans le programme principal en affichant le résultat de ce calcul.

Homogénéisation des données

- On constate que sont regroupés sous label différent ‘United States’ et ‘USA’. Ecrivez une fonction qui modifie la liste D en remplaçant tous les attributs contenant la valeur United States par USA.

- On constate également que dans certains cas, la valeur de l’attribut MONTANT est erronée. On souhaite recalculer tous les montants à partir des valeurs des champs PRIX_UNITAIRE et QUANTITE. Ecrivez une fonction qui reçoit la liste D et recalcule ces valeurs.

  Attention : les valeurs contenues dans le dictionnaire sont de type chaîne de caractère. Il faut donc au préalable “traduire” le champ PRIX_UNITAIRE en “réel” à virgule flottante et le champ QUANTITE en entier pour pouvoir ensuite calculer la valeur de MONTANT.

  rappels :
  Traduction d’une chaîne de caractère `s` en entier : `n = int(s)`
  Traduction d’une chaîne de caractère `s` en réel : `x = float(s)`
  Plus généralement : `x = eval(s)`

Chiffre d’affaires par pays

Appelez de nouveau la fonction calculant le nombre de ventes par pays pour vérifier que tout fonctionne bien.
Ecrivez ensuite une fonction qui retourne un dictionnaire donnant *le chiffre d’affaires* (c’est à dire la somme des montants) par pays.

Sauvegarde des données

a. format csv
On souhaite dans un premier temps sauvegarder les données corrigées de la liste D dans un format identique à celui du fichier de départ (format csv).
On crée tout d’abord un nouveau fichier ouvert en écriture :

```
g = open('ventes_corrige.csv','w')
```

On utilise ensuite l’objet `writer` de la librairie csv pour enregistrer les données dans un format adapté:

```
Lw = csv.writer(g, delimiter = ",")
```

On utilise ensuite la méthode `writerow` permettant d’ajouter des lignes au fichier:

```
Lw.writerow(e)
```

(avec `e` liste de valeurs)
remarque :
Il faut tout d’abord sauvegarder la liste des attributs `attr` puis ensuite sauvegarder ligne par ligne les valeurs contenues dans D. Attention, la ligne `D[i]` étant un dictionnaire, il faut créer la liste `e` à partir de `D[i]`:

```
e = []
for j in range(m):
        e.append(d[attr[j]])
Lw.writerow(e)
```

N’oubliez pas de fermer le fichier à la fin de l’écriture (`g.close()`).

b. format pickle

Il est possible également de sauvegarder les données en une seule opération dans un format différent de csv, le format “pickle” (format de sauvegarde propre à Python). Dans ce cas, la structure de données est sauvegardée “telle quelle”, il n’y a pas besoin d’extraire les lignes.

```
import pickle
```

On commence par ouvrir un fichier en écriture.

```
h = open("ventes_corrige.pkl", "w")
```

On sauvegarde notre liste D en utilisant la méthode `dump` du module pickle

```
pickle.dump(D, h)
h.close()
```

Les données ont été sauvées! ouvrez ce fichier avec un éditeur de texte pour voir à quoi ressemble ce format.

Si on veut relire l'objet contenu dans "`ventes_corrige.pkl`", on commence par ouvrir le fichier:

```
f = open("ventes_corrige.pkl", "r")
D_sauvegarde = pickle.load(f)
f.close()
```