Travaux Pratiques, huitième séance

Le TP sera réalisé en Python.

1. Introduction

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. Téléchargez ce fichier.

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 (par exemple **localc**). Les données sont maintenant "rangées" en lignes et colonnes pour faciliter la lecture.

Notez bien l'emplacement du fichier ventes.csv dans l'arborescence de fichiers.

2. Lecture de fichier texte et traitement des données

Créez maintenant un projet Python à l'aide de l'éditeur Pycharm.



Lors de la première ouverture, pensez à bien sélectionner python 3.5 comme interpréteur par défaut.

Si ce n'est pas le cas, effectuez la modification dans les menus : **Files > settings > project interpreter**

Le but est de définir une série de fonctions permettant de manipuler les fichiers de données de type csv.

2.1 Ouverture de fichier :

Ecrire une fonction ouvre fichier permettant d'ouvrir un fichier quelconque.

- La fonction prend en argument : le nom du fichier
- retourne : le descripteur du fichier ouvert (s'il existe) ou None sinon.

Il est important de vérifier que l'opération d'ouverture s'effectue correctement avant de poursuivre le programme (nombreuses possibilités d'erreur : fichier effacé, erreur de nom, pas de droits de lecture,...). On utilise une instruction de test spécifique pour vérifier que l'ouverture du fichier s'est correctement effectuée, de type try... except... (essaye ... sinon ...) permettant de prévoir une action de secours lorsqu'une une opération "risquée" échoue.



```
try:
    f = open(nom_fichier,'r')
except IOError:
    print("Erreur d'ouverture!")
```

• Créez le fichier de tests permettant de tester la fonction et testez-la!

2.2 Récupération de la liste des attributs :

La première ligne du fichier est différente des autres lignes. Il s'agit d'une 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.)

Ecrire une fonction get_attributs qui extrait du fichier csv la liste des attributs.

- la fonction prend en argument : le nom du fichier
- la fonction retourne :
 - la liste des attributs
 - le nombre m d'attributs

La fonction doit lire la première ligne du fichier uniquement.

- utiliser ouvre_fichier pour obtenir le decripteur de fichier f
- lire la première ligne à l'aide de la commande readline
- découper la chaîne de caractères à l'aide de la commande split
- Une fois nos manipulations sur le fichier effectué, on ferme le fichier avec la commande f.close().



- La commande s = f.readline() lit une ligne du fichier et la copie dans la variable s. Chaque nouvel appel à la fonction readline permet de lire une nouvelle ligne (jusqu'à ce qu'on atteigne la fin du fichier)
- La commande s.split(',') permet de construire une liste à partir d'une chaîne de caractères s et un caractère séparateur,
- Une fois la fonction écrite, testez-la à l'aide du fichier de tests!

2.3 lecture d'un tuple :

Ecrire une fonction get_premier_tuple qui extrait le premier tuple du fichier de données

- la fonction prend en argument : le nom du fichier
- la fonction retourne :
 - une liste de valeurs (un tuple)

La fonction doit lire la **deuxième** ligne du fichier uniquement.

- utiliser ouvre_fichier pour obtenir le descripteur de fichier
- appliquer 2 fois la commande readline afin d'obtenir la deuxième ligne
- découper la chaîne de caractères à l'aide de la commande split
- Testez votre fonction dans le fichier de tests. Vérifiez en particulier que la liste de valeurs contient bien le nombre d'éléments m attendu (calculé à l'aide de la fonction précédente).

2.4 Dictionnaire

Ecrire une fonction cree_dictionnaire:

- qui prend en argument une liste d'attributs attr et d'une liste de valeurs val,
- et retourne un dictionnaire d tel que pour tout j ∈ 0,...,m-1: d[attr[j]] = val[j]
- la fonction doit vérifier que les 2 listes fournies sont de même taille.
- Vous devez tester cette fonction, en particulier:
 - vérifiez que la liste des clés du dictionnaire correspond à la liste des attributs fournis
 - o vérifiez que la liste des valeurs du dictionnaire correspond à la liste des valeurs fournies
- une fois le dictionnaire construit, affichez quelques valeurs extraites du dictionnaire par exemple:

```
print('nom du client :' + d['NOM_CONTACT'])
print('montant de la vente :' + d['MONTANT'])
```

2.5 Extraction de la totalité du fichier :

Il est possible de lire 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.

Ecrire une fonction get all tuples qui:

- prend en argument : le nom du fichier
- retourne :
 - une liste d'attributs
 - le nombre m d'attributs
 - une liste de tuples (un tuple par ligne du fichier hors première ligne)
 - ∘ le nombre n de tuples



• Pour charger la totalité du fichier en une opération, utiliser :

```
liste_lignes = f.readlines()
```

- Tester cette fonction à l'aide du fichier de tests. Vous testerez en particulier que le nombre d'éléments est correct pour chaque tuple de la liste de tuples.
- Comment faire si ce n'est pas le cas?

3 - Librairie csv:

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

```
import csv
```

Pour ouvrir mon_fichier.csv:

```
flux_de_tuples = csv.reader(open("ventes.csv","r"))
```

• Remarque : l'objet flux_de_tuples est un *itérateur* qui se comporte comme un flux (on ne peut pas accéder directement au ième élément flux de tuples[i]).

Pour lire un enregistrement, on utilise

```
t = next(flux_de_tuples)
```

où t correspond à un tuple du fichier

Pour parcourir tous les éléments :

```
for t in flux_de_tuples:
...
```

- Réécrivez les fonctions de la partie précédente en utilisant maintenant les fonction de lecture de la librairie csv (au lieu de readline et split) et testez-les!
- Testez en particulier que le nombre d'éléments est correct pour chaque tuple produit par la fonction get_all_tuples.

4 - Tableau de dictionnaires

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

- Ecrire et tester une fonction cree_liste_dict
 - qui prend en argument : la liste d'attributs et la liste des tuples produites par la fonction get all tuples

• et retourne une liste de dictionnaires liste_dict

Extraire des valeurs

On souhaite extraire des listes de valeurs particulières à partir de liste_dict.

Si on affiche, par exemple, pour i de 0 à n-1, liste_dict[i]['PAYS'],on obtient une liste de pays avec de nombreux doublons.

On souhaite créer une liste de pays liste_pays sans doublon. Pour tester si un pays p est déjà dans la liste, on utilise :

if p in liste_pays:

- Ecrire et tester une fonction get pays
 - qui prend en argument liste dict obtenue à la question précédente
 - et retourne la liste des pays.
- Affichez cette liste.
- Ecrire et tester de même :
 - une fonction get produits pour obtenir la liste des produits
 - une fonction get clients pour obtenir la liste des clients.

Statistiques basiques

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

- a -

- Ecrire et tester une fonction ventes_france qui prend en argument la liste liste_dict et retourne le nombre de ventes effectuées en France.
- Affichez ce résultat dans le programme principal.

- b -

• Ecrire et tester une fonction ventes_par_pays qui retourne le nombre de ventes par pays. Cette fonction prend en argument la liste liste_dict et la liste des pays et retourne un dictionnaire nb_ventes contenant le nombre de ventes par pays.

Pour construire ce dictionnaire,

- on crée tout d'abord un dictionnaire vide nb_ventes = {}
- On parcourt ensuite la liste des pays et on initialise chaque entrée du dictionnaire :



```
for pays in liste_pays:
    nb_ventes[pays] = 0
```

• puis on parcourt les éléments de liste_dict : pour chaque élément de liste_dict, on incrémente le nombre de ventes correspondant à



liste_dict[i]['PAYS']

- Testez cette fonction
- Dans le programme principal, affichez le résultat de ce calcul.

Homogénéisation des données

On constate que sont regroupés sous label différent 'United States' et 'USA'.

Écrivez et testez une fonction nettoie USA:

• qui modifie liste_dict 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 0UANTITE.

Écrivez et testez une fonction nettoie montant

- qui reçoit une liste_dict
- et recalcule ses 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 et testez une fonction calcule_CA qui retourne un dictionnaire donnant le chiffre d'affaires (c'est à dire la somme des montants) par pays.

5 - 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:

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

• On utilise ensuite la méthode writerow permettant d'ajouter des lignes au fichier:

```
flux_de_sortie.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 liste_dict. Attention, la ligne liste_dict[i] étant un dictionnaire, il faut créer la liste e à partir de liste_dict[i]:



```
d = liste_dict[i]
e = [d[attr[j]] for j in range(m)]
flux_de_sortie.writerow(e)
```

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

b. format json

Il est possible également de sauvegarder les données en une seule opération dans des formats différent de csv. Le format json correspond à une mise en forme de type dictionnaire, facile à lire et interpréter.



voir : données structuréesvoir : données hiérarchisées

Pour importer la librairie json :

import json

On commence par ouvrir un fichier en écriture.

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

On sauvegarde liste_dict en utilisant la méthode dump du module json

```
json.dump(liste_dict, h, sort_keys=True, indent=4)
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.json, on commence par ouvrir le fichier:

```
f = open("ventes_corrige.json", "r")
liste_dict_new = json.load(f)
f.close()
```

From:

https://wiki.centrale-med.fr/informatique/ - WiKi informatique

Permanent link:

https://wiki.centrale-med.fr/informatique/tc_info:tp8

Last update: 2018/12/18 09:18

