# Travaux Pratiques, première séance

Le TP sera réalisé en Python.

Normalement, vous devez connaître le langage python, l'environnement Pycharm & le développement par les tests. Si ce n'est pas le cas, les tutos sont pour vous...

#### 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

#### **Ouverture du fichier en Python:**

Créez maintenant un script 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** 

Dans le script, ouvrez le fichier avec la commande:

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

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('monfichier.dat','r')
except IOError:
    print("Erreur d'ouverture!")
```

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)

#### Récupération de la liste des attributs :

La première ligne du fichier 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 s en utilisant la commande split. (la commande s.split(',') permet de construire une liste à partir d'une chaîne de caractères s et un caractère séparateur ,)

Définissez la variable m contenant le nombre d'attributs.

#### Extraction d'une ligne :

Les lignes suivantes contiennent des valeurs d'attributs.

Lisez une nouvelle ligne et affichez-la.

Chaque ligne doit être "découpée" pour extraire les différentes valeurs.

- Définissez une liste de valeurs val à partir de la nouvelle ligne en utilisant la commande split.
- Affichez la liste val. Vérifiez que val contient bien m éléments.

#### **Dictionnaire**

 Ecrivez une fonction qui retourne un dictionnaire d à partir de attr et val tel que pour tout j  $\in 0, \dots, m-1: d[attr[j]] = val[j]$ 

 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'])
```

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

#### 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.

• Ouvrez de nouveau le fichier, et stockez la totalité du fichier dans une liste liste lignes.

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

- Définissez une variable n contenant le nombre d'éléments de cette liste.
- Créez un dictionnaire correspondant à la ligne d'indice 7 : liste\_lignes[7]. Il y a un problème!! lequel?

#### 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 se comporte comme un flux (on ne peut pas accéder directement au ième élément flux de tuples[i]).

Pour lire un enregistrement, on utilise

```
e = next(flux_de_tuples)
```

- Le premier élément lu e correspond à une liste d'attributs.
- Affichez le contenu de e puis copiez-le dans une liste attr.

Pour afficher tous les éléments :

```
for e in flux_de_tuples:
    print(e)
```

avec e correspondant maintenant à des listes de valeurs (celles à partir de la ligne 1)

• Reprenez les opérations vues dans la partie précédente :

- construction d'un dictionnaire pour la 1ère ligne, puis pour la 7ème ligne.
- quelle(s) différence(s) avec le cas précédent?

## 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))

Pour parcourir la totalité de la liste flux\_de\_tuples, on peut écrire :

```
for e in flux_de_tuples:
    ...
    for i in ...
        d[attr[i]] = e[i]
    ...
```

Ecrire une fonction qui prend en argument le descripteur flux de tuples et retourne 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:
```

- Ecrivez une fonction qui prend en argument liste dict et retourne la liste des pays.
- Affichez cette liste.
- Faites de même pour la liste des produits et 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 -

- Tout d'abord, écrivez une fonction 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 ensuite une fonction 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 en affichant le résultat de ce calcul.

## Homogénéisation des données

On constate que sont regroupés sous label différent 'United States' et 'USA'. Écrivez une fonction 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 QUANTITE. Écrivez une fonction qui reçoit la liste\_dict 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.

## 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 structurées
- voir : 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/restricted:tc-d:tp1:travaux\_pratiques\_premiere\_seance\_2017

Last update: 2018/03/25 22:24

