T02-02 Traitement des données

Partie 1 : créer un fichier de données, le modifier, le lire: la BASE!

1- Les instructions de base

a- conseil de sioux: **créer un répertoire courant de travail à accès facile** pour le fichier à travailler:

```
import os  #on importe le module nécessaire pour travailler les fichiers et dossiers(=répertoires)
os.chdir("C:/tests python")  # indique le répertoire existant de travail courant 'tests pythons' à la racine de C:
print(os.getcwd()) # pour vérifier qu'on a bien changé le répertoire de travail courant...
```

b- Pour **nommer un fichier** dans python, il faut décrire le **chemin absolu** pour s'y rendre:
exemple C:/mon_dossier/prog_python/mon_fichier.txt

c- Il faut toujours ouvrir le fichier selon un mode d'ouverture particulier(lecture 'r', écriture 'w', ajout 'a'):

```
mon_fichier = open("fichier.txt", "r") # affecte le fichier ouvert à une variable mon_fichier pour la LIRE
print(mon_fichier) # pour indiquer les caractéristiques de ce fichier
<_io.TextIOWrapper name='fichier.txt' encoding='cp1252'>
```

Rq: **'r'**: le fichier doit être existant, **'w'** le contenu du fichier est écrasé, **'a'** le contenu n'est pas écrasé, on peut le modifier. Dans les cas de **'w' et 'a',** si le fichier n'existe pas, il est créé.

On peut ensuite affecter le contenu de notre fichier à une variable, l'afficher, la modifier...

```
contenu = mon_fichier.read() # affecte à la variable contenu le contenu du fichier
print(contenu) # affiche le contenu du fichier(en fait de la variable mais c'est la même chose ici…)
mon_fichier.close() # pour fermer le fichier et le rendre disponible aux autres applications(indispensable!!!).
```

Alternative plus rapide mais moins pédagogique...:

```
with open('fichier.txt', 'r') as mon_fichier:
        texte = mon_fichier.read()
        print(texte)
```

→ complétez votre mémo python des nouvelles instructions utilisées....
open("fichier.txt", "r"), open("fichier.txt", "w"), open("fichier.txt", "a") ; close("fichier.txt") ; with....
la façon d'adresser un fichier F:/dossier1/dossier1,1/dossier1,1,1/fichier.txt

2- Exercice d'application:

En utilisant l'interface python:
- Dans un dossier "mes fichiers python" situé à la racine de votre clef, créer un fichier .txt vide
- afficher son contenu
- le fermer(vous essaierez ensuite sans le fermer pour constater le conflit...).
- ouvrez le avec un traitement de texte... et inscrivez y votre NOM et votre prénom et un mot secret, enregistrez, fermez le.
- afficher son contenu

Partie 2: opérations sur les données

1, Quelques opérations de base à effectuer sur les données.

1- Un exemple: créer un fichier texte liste de course....

Ressources:

liste.write(variable à ajouter) #ajoute la variable dans le fichier 'liste', ici ce sera une chaîne de caractères(str)
liste.write('\n') #va à la ligne
l'instruction '{}'.format()
→ print('{}, c'est à présent ton essai n{}'.format(NOM, i)) # permet d'afficher
JEAN RENE, c'est à présent ton essai n°5 #*si NOM= JEAN RENE et i = 5....*

2- Exercice d'application:

En utilisant l'interface python:

Niveau 1:

- Créer une programme python qui vous permettra de dresser une liste de course lisible enregistré dans un fichier 'listedecourse.txt'
- ouvrir le fichier 'liste de course'

Niveau 2:

- Créer une programme python qui demande à l'utilisateur:
  - combien de produits il désire mettre sur sa liste de course
  - quels sont ces produits
  - et que ce programme crée un fichier 'liste de course.txt' dans lequel on trouve la liste.
- ouvrir le fichier 'liste de course'

Niveau ultime:

Niveau 2 + le programme doit dresser la liste afin que les produits soit numérotés alors que l'utilisateur n'a entré que le nombre de produits total et le nom de chacun.

exemple de rendu au sein du fichier quel que soit le niveau:

Ma liste de course
1- œufs
2- viande
3- un arbuste pour le jardin

A faire:

On peut donc créer un fichier qui renferme des données afin de les conserver pour les consulter(read) ou les modifier(append) --> Allez jeter un coup d'œil au fichier créé, complétez votre mémo python des nouvelles instructions utilisées....

fichier.write(' ')
fichier.read()
print('{}e de la course mixte et {} garçon'.format(position mixte, position garçon))

2- Créer et modifier des listes/tables: (HProg)

1-Très utile pour stocker les bases de données, on peut créer des tables, les variables vont donc être définies par des descripteurs. Le plus compatible est fichier de type csv.

Ressources:

Liste=[variable1, variable2, variable3] # on affecte à Liste une liste de trois variables.

Si on a une chaîne de caractères de type chaîne='NOM, PRENOM, AGE'

Liste=chaine.split(',') # permet de transformer la chaîne en liste de 3 variables séparées par des virgules.

ListeTot=[[liste1],[liste2],[liste3]] # permet de créer une liste de liste et de générer un tableau lisible par un tableur.

1. Variable=m[j][i] #ressort l'élément i dans la liste j de la liste de liste m.
2. Grrrrrr.insert(*i, x*) #Insère x en i dans la liste Grrrrrr, ajoute un élément dans la liste à une position précise(décale le reste!!!)
3.

Pour créer un fichier csv, c'est comme pour un txt mais il faut réfléchir à la structure:

a. lors de la lecture, chaque ligne pourra être transformée une liste par python à condition de séparer les éléments par un caractère précis(une virgule, un point virgule, une tabulation…), on se servira de la fonction split(« ? »)

b. penser, si l'on veut représenter un tableau, à renseigner la première ligne comme intitulés de colonnes.

c. Il faut que chaque ligne possède le même nombre d'éléments et qu'ils soient rangés dans le même ordre que les intitulés.

Un exemple:

```
f=open('test.csv', 'w') # lien vers un fichier on l'affecte la variable f en écriture, on écrase le contenu
f.write('intVar1,intVar2,intVar3\n') #on crée les intitulés et on va à la ligne
f.close() #on ferme dans la config 'w'

f=open('test.csv', 'a') # lien vers un fichier on l'affecte la variable f en ajout, on écrase pas
4. f.write(var1+','+var2+','+var3+'\n') #on ajoute à la liste 3 valeurs de variables var1 à 3 et on va à la ligne
f.close() #enferme dans la config 'a'
```

2- Exercice d'application:

Niveau 1:

Créer à présent un programme permettant de créer, en interaction avec un utilisateur, une base de donnée concernant ce client.

Le client devra renseigner son nom, prénom, son sexe(M ou F), son age, son adresse mail, sa pointure de chaussure. Le fichier se nommera basededonneesclients.csv.

Niveau 2:

Créer à présent un programme permettant de créer, en interaction avec X(à demander) utilisateurs successifs, une base de donnée clients.

Chaque client devra renseigner son nom, prénom, son sexe(M ou F), son age, son adresse mail, sa pointure de chaussure. Le fichier se nommera basededonneesclients.csv.

3- Rechercher, filtrer, trier ou calculer sur une ou plusieurs tables des listes/tables:

1- La **recherche dans des données structurées** a d'abord été effectuée selon une **indexation préalable faite par l'homme** lorsqu'il crée ses listes ou tables. Des algorithmes peuvent ensuite d'automatiser l'indexation à partir de textes, d'images ou de sons. Une table de données peut faire l'objet de différentes opérations: rechercher une information précise dans la collection grâce aux descripteurs, trier la collection sur une ou plusieurs propriétés, filtrer la collection selon un ou plusieurs tests sur les valeurs des descripteurs, effectuer des calculs, mettre en forme les informations produites pour une visualisation par les utilisateurs.

Objectifs du projet: lire un fichier csv afin de trier son contenu et d'utiliser une partie des données qu'il comporte.

Ressources : *(tout n'est pas forcément utile...)*
votre mémo Python.
list.append(*x*) **#**Ajoute un élément à la fin de la liste 'list'
list.insert(*i*, *x*) **#**Insère un élément à la position indiquée. Le premier argument est la position de l'élément courant avant lequel l'insertion doit s'effectuer
list.remove(*x*) **#**Supprime de la liste le premier élément dont la valeur est égale à *x*
list.clear() **#**Supprime tous les éléments de la liste
list.index(*x*[, *start*[, *end*]]) **#**Renvoie la position du premier élément de la liste dont la valeur égale *x*
list.count(*x*) **#**Renvoie le nombre d'éléments ayant la valeur *x* dans la liste.

2- Exercice d'application:

Nous travaillerons sur le fichier nat2019.csv qui recense tous les prénoms nés en France, par année depuis 1900.
Votre programme python devra demander le prénom que vous recherchez et afficher les lignes correspondantes au prénom ainsi que la somme de toutes les naissances sous ce prénom depuis 1900.

Aide Pas à pas:
a- On l'invente pas, ce fichier nécessite un traducteur, codec, commencez le programme ainsi:
import codecs # le fichier nécessite des traducteurs, il est codé en UTF-8
monfichier=codecs.open('K:/nat2018.csv','r','utf-8') #on affecte la variable 'monfichier' du fichier à travailler
b- Il va falloir séparer cette immense chaîne de caractères en listes: d'abord une liste par ligne, puis pour chaque ligne, la transformer en liste qui comprend autant de variables que de colonnes.

c- Il va falloir faire autant de boucles qu'il y a de lignes(moins l'entête des intitulés)

Mon programme fait 14 lignes et fonctionne à merveille, alors courage !!!