TP 2: LA GESTION DES FICHIERS ET DES BASES DE DONNÉES EN PYTHON

Exercice 1

L'exercice consiste à créer une fonction `nbrLignes()`, qui a pour paramètre le *nom d'un fichier* `texte` et qui renvoie le nombre de lignes de ce fichier.

Exercice 2

Dans une société multinationale, on reçoit plusieurs candidatures. Pour bien mener la campagne de recrutement, on souhaite gérer un fichier typé et intitulé **"concours.txt"** qui comporte les enregistrements relatifs aux candidats d'un concours. Chaque enregistrement est composé de: `ncin`, `nom`, `prenom`, `age`, et `decision`, sachant que `decision` est un type contenant les identificateurs suivants: **"ad"** (**"admis(e)"**), **"r "** (**"refusé(e)"**), **"aj"** (**"ajourné(e)"**), et les différentes coordonnées sont séparées par un point-virgule (**";"**).

1. Définir la fonction `saisir()` qui permet de remplir les données relatives aux candidats dans le fichier **"concours.txt"** en contrôlant la saisie de la décision d'ajouter, à nouveau, un candidat.
2. Définir la fonction `admis()` qui permet créer le fichier **"admis.txt"** comportant les données relatives aux candidats admis.
3. Afin d'aider à la prise de décision, la société souhaite sélectionner, en priorité, les candidats admis et âgés de moins de 30 ans. De ce fait, créer la fonction `attente()` qui produira à partir du fichier **"admis.txt"**, un nouveau fichier intitulé **"attente.txt"** comportant les données relatives aux candidats admis et âgés de plus de 30 ans. Une ligne du fichier **"attente.txt"** comprend le `ncin`, le `nom`, et le `prenom` d'un(e) candidat(e), et les coordonnées sont séparées par un point-virgule (**";"**).
4. Définir la fonction `statistiques(dec)` qui permet de calculer le pourcentage des candidats pour la décision `dec` (**"admis(e)"**, **"refusé(e)"**, **"ajourné(e)"**).

Par exemple, pour les candidats admis, nous aurons:

$$Le\ pourcentage\ des\ candidats\ admis = \left(\frac{Nombre\ \ de\ \ candidats\ \ admis}{Nombre\ \ de\ \ candidats}\right) * 100$$

5. Définir la fonction `supprimer()` qui supprimera, du fichier **"admis.txt"**, les candidats âgés de plus de 30 ans.

N.B: On suppose que les fichiers seront mis à la racine du lecteur D: (ou C:)

Exercice 3

1. Créer un fichier CSV `"elections.csv"` contenant tous les candidats d'une campagne électorale, trié(e)s par ordre croissant sur le nom.

   Chaque ligne a les colonnes suivantes:
   - `Nom`: nom du candidat.
   - `Prenom`: prénom du candidat.
   - `Age`: l'âge du candidat.
   - `Nombre de voix`: Nombre de voix obtenues.

2. Écrire le code Python qui lit le fichier CSV et qui écrit :

   - Le nombre de candidats ayant plus que 10 000 voix.
   - Le nombre de candidats âgés de moins de 40 ans.
   - La moyenne de voix obtenues.
   - Le gagnant dans cette campagne.
   - Le candidat ayant le moins de voix.

3. Écrire une fonction qui trie les résultats issus du fichier `"elections.csv"` suivant le nombre de voix par ordre décroissant et qui sauvegarde le résultat dans un nouveau fichier CSV `"resultatsElections.csv"`.

Exercice 4

Un opérateur téléphonique propose de créer un répertoire téléphonique contenant les noms des personnes et leurs numéros de téléphones.

Ce répertoire est représenté par un fichier texte nommé `"contacts.txt"`. Ce fichier contiendra des objets de type `Contact`. Ce dernier est caractérisé par un `nom` et le `numéro de téléphone`.

1. Écrire la procédure `creerFichier()` qui permet de créer ce fichier en utilisant le package `Pickle` contentant `N` contacts.

2. Écrire la procédure `ajouterContact(nom, telephone)` qui ajoute les données d'une personne (`nom, telephone`) sur un enregistrement à la fin du fichier `"contacts.txt"`. Si le numéro de téléphone ne contient pas huit chiffres, un message d'erreur sera affiché.

3. Écrire la fonction `recherche(nom)` qui renvoi le numéro de la personne dont le nom est passé en paramètre. Si le nom de la personne n'existe pas dans le répertoire alors la fonction renvoi la constante `None` (rien).

4. Écrire la fonction `modifierContact(nom,num)` qui modifie le numéro du contact dont le nom est transmis en paramètre. Cette fonction retourne `True` si la modification est effectuée correctement, sinon elle retourne `False`.

5. Écrire la procédure `copie()` qui crée une copie du fichier `"contacts.txt"`.

6. Écrire la procédure `affiche(nomFich)` qui affiche à l'écran (objet par objet) les données de tous les contacts contenus dans le fichier `"nomFich.txt"`.

7. **(À rendre)** Dans un répertoire téléphonique, un numéro de téléphone doit exister une seule fois, on suppose que, par erreur, nous avons répété quelques numéros sous des noms différents. Écrire la procédure `suppDoublants(nomFich)`, qui affiche, à l'écran, les numéros en double avant de les supprimer.

8. **(À rendre)** Écrire la procédure `trier()` qui permet de trier les contacts du fichier `"contacts.txt"` selon l'ordre alphabétique des noms.