I Accès à un fichier en python

Chemins

Il existe deux manières d'indiquer la position exacte d'un fichier :

- son chemin absolu, qui commence sous windows par une lettre de lecteur.
  Par exemple C:\Users\moi\Documents\TD14.pdf
- son chemin relatif, c'est à dire son emplacement par rapport au *dossier de travail courant*. Ce dossier courant est déterminé au lancement du programme (quand on lance python, il s'agit au départ du dossier contenant le programme python.exe) et peut être modifié à tout moment.

```
import os
os.getcwd()  # retourne, sous forme d'une chaîne de caractère
# le nom du dossier courant
```

```
os.chdir('C:\\Users')  # change le dossier courant
```

Exercice 1
Dans Spyder, ouvrez le fichier **td14.py** et exécutez le en utilisant F5 ou la flèche verte. Observez attentivement la ligne `runfile` qui vient d'apparaître dans la console. Elle contient

- le chemin absolu vers votre script python
- le changement de dossier de travail (*working directory*, ou encore *wdir*)

Vérifier maintenant quel est le dossier (ou répertoire) courant et que ce dossier contient bien nos deux fichiers de données (**aeroports** et **pays**).
Ainsi le chemin relatif de ces fichiers est très simple : il s'agit seulement du nom.

Lecture en python

On peut lire en python le contenu de n'importe quel fichier (à condition d'avoir les droits de lecture sur le fichier en question) par la syntaxe

```
f = open(chemin, 'r', encoding='utf-8')
# puis au choix :
texte = f.read() # on récupère le contenu en entier
# ou si on préfère
lignes = f.readlines()  # on récupère la liste des lignes
# et enfin, à ne pas oublier
f.close()
```

Il faut choisir le mode de lecture (texte entier ou par lignes) : si vous exécutez les deux commandes de lecture à la suite, la variable `lignes` contiendra une liste vide car le fichier aura déjà été lu.
Pour éviter d'avoir à fermer le fichier manuellement (et même s'il y a une erreur le fichier sera correctement fermé) :

```
with open(chemin, 'r', encoding='utf-8') as f:
    texte = ...
    # ou
    lignes = ...
# ici le fichier est bien fermé et on peut travailler avec les données lues.
```

II Application

Ouvrir dans Spyder le fichier **pays.csv** et trouver le format utilisé pour écrire les données (comprendre la structure du fichier)

Exercice 2
Nous allons utiliser la lecture des fichiers ligne par ligne. Le premier problème rencontré est le caractère de fin de ligne '\n'. Il est présent à la fin de chaque ligne, sauf la dernière, mais n'a pas de sens par rapport à nos données.

```
ligne = 'code;nom\n'  # il s'agit de la première ligne de notre fichier
ligne.strip()  # retire les espaces, saut de ligne et tabulation situés au début où à la fin
# on a créé une nouvelle chaîne, sans le caractère de fin de ligne.
```

Compléter la fonction `lignes_fichier` qui prend comme argument un chemin (une chaîne de caractères), et retourne la liste des lignes de ce fichier, sans caractère de fin de ligne.
Tester sous la forme

```
liste_lignes('pays.csv')  # remarquer les guillemets et le chemin relatif
```

Exercice 3
On remarque que la première ligne de notre fichier contient en fait les étiquettes des colonnes, et la suite du fichier consiste en une ligne par pays.
On souhaite obtenir comme résultat de la lecture du fichier **pays.csv** une liste de la forme

```
[ {'code': ..., 'nom': ...}, {...} ... ]
```

où chaque pays est représenté par un dictionnaire dont les clés sont les étiquettes (présentes en première ligne) et les valeurs données par chacune des autres lignes.
Nous allons d'abord nous pencher sur la création d'un dictionnaire dont on donne les clés et les valeurs.
Compléter la fonction `cre_dico(cles, valeurs)` qui prend comme argument deux listes de même longueur et crée le dictionnaire dont les clés sont les éléments de `cles` et les valeurs correspondantes les éléments de même indice dans `valeurs`.
Par exemple, si `d` est le dictionnaire créé, on aura

```
d[cles[0]] == valeurs[0]
```

Exercice 4
Compléter la fonction `liste_dico` qui prend en argument une liste de lignes nettoyées et retourne la liste des dictionnaires correspondants (voir la question précédente). Cette fonction doit pouvoir travailler sur tout fichier dont le format est similaire (par exemple, sur le fichier **aeroports.csv**).
Une aide technique sur les chaînes :

```
s = 'code;nom'
parties = s.split(';') # sépare la chaîne suivant les ; et retourne la liste de toutes les parties ainsi séparées.
# ici parties est la liste ['code', 'nom']
```

III Manipulation et analyse des données

Les dictionnaires peuvent, comme les listes, être créés par compréhension. Il s'agit là d'une technique avancée. La syntaxe est :

```
{cle : val for ....}
```

Exercice 5
Créer dans la console le dictionnaire dont les clés sont les codes des pays et les valeurs leurs noms. Ce dictionnaire nous servira à traduire le contenu du fichier **aeroports.csv**

Exercice 6
Créer dans la console la liste des aéroports, chacun étant représenté par un dictionnaire, et en remplaçant le code du pays par son nom.

Exercice 7
Trouver le continent où se situent le plus d'aéroports. Même question avec les pays.

En cas de besoin, pour un dictionnaire `d`, on peut accéder à la "liste" de ses clés par `d.keys()` et à ses valeurs par `d.values()`.
En toute rigueur, les objets retournés ne sont pas des listes python, car ces objets sont non modifiables et on ne peut pas accéder à leurs éléments par indice, mais ils peuvent servir dans une boucle for sans problème.

Exercice 8
On souhaite illustrer la question précédente par un diagramme. Consulter l'aide de la fonction `pie`

```
import matplotlib.pyplot as plt  # à exécuter une fois pour toute dans la console
help(plt.pie)  # affiche, dans la console, l'aide associée à une fonction
```

Tracer le diagramme de répartition par continent, en utilisant des étiquettes raisonnables.

Exercice 9
Construire la liste des aéroports se situant en France.

Exercice 10
Construire le dictionnaire dont les clés sont les noms de pays et les valeurs la liste des aéroports se situant dans ce pays.

Exercice 11
Trouver le nombre moyen d'aéroport par pays, l'écart type, le nombre médian.