I Lecture de fichier texte

1.1 Découverte de la structure

Exercice 1
Ouvrez le fichier *pays.csv* dans l'éditeur de texte de spyder et expliciter la structure de chaque ligne. Il s'agit de comprendre quelles sont les données présentes et comment elles sont séparées les unes des autres.

Exercice 2
A l'aide de la console et de l'aide python, comprendre les fonctions `s.strip` et `s.replace` quand `s` est une chaîne de caractère. On pourra utiliser comme exemple (RAPPEL : les copier-coller depuis un PDF peuvent être piégeux. Mieux vaut recopier les commandes)

```
s = '  Voici une chaîne de caractère. '
# essayer :
s.strip()
s.strip(' .')
s.strip(' .V')
# et aussi
s.replace(' ', '-')
help(s.replace)
```

Même question avec `s.split` qui prend comme argument une chaîne de caractères. On pourra tester `s.split(' ')`, `s.split('a')`

1.1.1 Lire un fichier en python

Toutes les méthodes que nous rencontrerons commencent par

```
f = open('une chaîne qui est le chemin vers le fichier', encoding='utf-8')
```

Chemins : le chemin menant à un fichier peut être absolu (en commençant à une racine du système de fichier) comme par exemple "C :
Programmes
WinPython
Spyder.exe" ou relatif comme "Spyder.exe". Dans le deuxième cas on fait en fait référence au **répertoire courant** qui varie suivant le programme en cours d'exécution et l'instant. Pour connaître le répertoire courant (working directory en anglais) en python :

```
import os # un import s'effectue une fois pour toute dans la console
os.getcwd() # get current working directory
```

Pour changer le répertoire courant et donc pouvoir utiliser un chemin de fichier relatif dans la commande `open`, on peu utiliser deux méthodes :

1. Le dire directement à python

   ```
   os.chdir('chemin vers le nouveau répertoire courant')
   ```

2. Exécuter toute la feuille avec spyder (on voit apparaître un `runfile` dans la console) via F5, le menu "Run" ou la flèche verte. Dans ce cas le nouveau répertoire courant devient le dossier contenant le script que l'on vient d'exécuter (visible dans la partie `wdir=` de la commande `runfile`)

Exercice 3
Réussir à ouvrir le fichier **pays.csv** et créer ainsi la variable `f`.

1.1.2 Accès aux caractères

Les fichiers que nous allons lire sont composés de texte. Nous voulons lire ce texte grâce à python (en utilisant la bonne table de caractère, d'où le paramètre `encoding=` passé à la fonction `open`).
Il y a encore une fois plusieurs méthodes :

1. On peut vouloir lire tout le fichier d'un coup :

   ```
   texte = f.read()
   ```

2. Ou alors accéder aux différentes lignes du texte :

```
lignes = f.readlines()
```

3. Une dernière méthode, pour travailler avec les lignes une par une :

```
for ligne in f:
    ....
```

car `f` (l'objet python représentant notre fichier) est considéré comme un conteneur de lignes.

Dans tous les cas, une fois les données du fichier lues, il faut absolument utiliser la commande

```
f.close()
```

qui va libérer les ressources (nombre de fichier ouvert en même temps) du système d'exploitation.

Exercice 4
A l'aide des outils présentés dans cette section, extraire la deuxième ligne de notre fichier **pays.csv** et séparer les trois données contenues. On veut obtenir une chaîne de caractère pour le nom et des entiers pour les populations et superficies. Pour cela on pourra utiliser la fonction `int`

1.2 Extraction des données

Exercice 5
Créer une fonction **lit_pays()** qui retourne 3 listes : celle des noms de pays, celle des populations et celle des superficie.
On n'oubliera pas de fermer le fichier ouvert après lecture.
Pour mémoire, on peut créer une liste vide, puis lui ajouter un élément par la syntaxe

```
liste_nom = []
...
nom_pays = ...
liste_noms.append(nom_pays) # ajoute la valeur contenue dans la variable nom_pays à la fin de la liste
```

Exercice 6
Créer la liste des densités de population (en habitant/kmš) des pays étudiés, dans l'ordre d'apparition du fichier.

Exercice 7
Trouver le pays ayant la plus forte et celui ayant la plus faible densité de population et afficher leurs noms.

II Graphiques

2.1 Rappels sur l'utilisation

```
import matplolib.pyplot as plt
import numpy as np
```

La création de courbes se fait de la manière suivante : On considère deux listes X et Y de même longueur n

```
plt.plot(X, Y)
```

trace la ligne brisée qui relies tous les points de coordonnées $(\mathrm{X}[i], \mathrm{Y}[i])$ pour i entre 0 et $n-1$

Exercice 8
Tracer la fonction arctan sur $[-5, 5]$. On prendra une liste d'abscisses de longueur 200 (cf `np.linspace`) et on pourra utiliser la fonction `np.arctan`.

Exercice 9
Créer la liste `pop` telle que `pop[i]` contient le nombre de pays ayant une population (en millions) dans $[20i, 20(i+1)[$. Créer également la liste `abscisses` des $20 * (i+1)$ pour toutes les valeurs de i dont vous aurez besoin pour `pop` et afficher le graphique correspondant.
Comparer à

```
plt.hist(pop, abscisses)
```

III Ecriture

Exercice 10
Ecrire dans le fichier *densité.csv* les données : nom du pays ; densité.
On pourra au choix donner des titres aux colonnes, ou non, dans la première ligne.
Pour écrire dans un fichier situé à l'adresse `chemin` :

```
f = open(chemin, 'w')
...
f.write(ici mettre une chaine de caractères à écrire dans le fichier)
...

f.close()
```

Pour passer à la ligne, on utilise le caractère spécial $\backslash n$

Exercice 11
Ecrire une fonction **combine(L1, L2, L3)** qui prend 3 listes de même longueurs comme arguments et retourne la liste $[[\mathrm{L1}[0], \mathrm{L2}[0], \mathrm{L3}[0]], \dots]$ où chaque éléments est une liste de longueur 3.

Exercice 12
Expérimenter la fonction

```
L.sort()
```

où L est une liste construite avec la fonction **combine**. On fera au moins 3 expériences pour 3 valeurs pertinentes de L1.

Exercice 13
Créer les fichiers csv *pays-par-population.csv* et *pays-par-superficie.csv* qui contiennent la liste de nos pays triées dans l'ordre croissant suivant leur titres...

IV Bonus

Trouver comment tracer un cercle de centre donné et rayon donné, comment tracer un polygone régulier à n côtés, une droite d'équation $y = ax + b$ (on donne a, b), cette même droite mais limitée à un rectangle donné (comment se donner un rectangle le plus efficacement possible ?).