Consigne : vous devez rendre une copie double avec vos réponses aux questions et tous les codes demandés écrits à la main en respectant scrupuleusement l'indentation ainsi qu'un fichier source `Python` *contenant tout le code produit.*

Méthode *Lecture / écriture de fichiers textes*

En Python, l'accès à un fichier texte se fait par l'intermédiaire d'un descripteur de fichier créé à l'aide de la primitive `open(nom,mode)`. Une fois que les manipulations sont terminées, il faut bien penser à fermer le descripteur de fichier avec `f.close()`.

Lors de l'ouverture d'un fichier on précise l'un des trois modes d'accès : lecture `'r'`, écriture `'w'` ou ajout `'a'`. Attention, pour l'ouverture en mode écriture, les modes `'w'` ou `'a'` créent le fichier s'il n'existe pas mais le mode `'w'` écrase le fichier s'il existe déjà.

Pour bien manipuler un fichier texte, il faut d'abord connaître la façon dont il est structuré!!!

Fonctions	Rôle
`f = open('nom_fichier.txt','w')`	accès en écriture avec création d'un nouveau fichier
`f = open('nom_fichier.txt','r')`	accès à un fichier existant en mode lecture
`f = open('nom_fichier.txt','a')`	accès en écriture à un fichier existant en mode ajout
`f.write('texte')`	ajout de 'texte' dans le fichier
`f.writelines(liste)`	ajout d'une liste de lignes dans un fichier
`f.read()`	lecture de tout le fichier
`f.read(8)`	lecture des 8 premiers caractères du fichier
`f.readline()`	lecture de la ligne courante du fichier
`f.readlines()`	lecture de toutes les lignes stockées dans une liste
`for ligne in f`	itération sur les lignes du fichier
`f.close()`	fermeture du fichier

Lecture de tout le fichier

```
f = open('fichier.txt','r')
data = f.read()
f.close()
```

Lecture ligne par ligne

```
f = open('fichier.txt','r')
for ligne in f:
    #traitement sur la ligne
f.close()
```

Lecture ligne par ligne

```
f = open('fichier.txt','r')
ligne = f.readline()
while ligne != '':
    #traitement sur la ligne
    ligne = f.readline()
f.close()
```

Capture dans une liste de toutes les lignes

```
f = open('fichier.txt','r')
listlignes = f.readlines()
f.close()
```

Écriture

```
f = open('fichier.txt','r')
f.write('Ligne ecrase tout\n')
f.close()
```

Ajout à la fin

```
f = open('fichier.txt','a')
f.write('Ligne de plus\n')
f.close()
```

Dans ce devoir, vous allez travailler sur le fichier texte `fr-esr-apb_voeux-et-admissions.csv` téléchargeable depuis cette page du site https://www.data.gouv.fr/.

Ce jeu de données ouvert (on parle d'open data), présente les vœux de poursuite d'études et de réorientation dans l'enseignement supérieur ainsi que les propositions des établissements pour chaque formation à la fin du processus d'affectation de la plateforme APB (précurseur de Parcoursup) pour les session 2016 et 2017.

1 Réglage du répertoire de travail

1. Dans un environnement de programmation `Python`, comme `Pyzo`, créer un nouveau fichier source `DM3-Eleve.py` où vous remplacez `Eleve` par votre nom. Ce fichier recevra tous les codes produits au cours de ce DM.

2. Régler le répertoire de travail sur le répertoire contenant le fichier `DM3-Eleve.py`. Dans `Pyzo` il suffit d'exécuter le programme avec la première option d'exécution `Exécuter en tant que script`, dans `Spyder` il n'y a rien à faire et dans `Idle` on peut suivre ce tutoriel.

2 Lecture de fichier

1. Saisir le code suivant qui permet de positionner le nom du fichier de données dans la variable globale `SOURCE` et de lire les quatre premières lignes du fichier.

```
SOURCE = 'fr-esr-apb_voeux-et-admissions.csv'

f = open(SOURCE)
for k in range(4):
    ligne = f.readline()
    print(ligne)
f.close()
```

 Il ne faut pas oublier de fermer le fichier lorsqu'on a terminé le traitement. En mode écriture, l'écriture effective de données ne se fera qu'après cette instruction et en mode lecture, si on ouvre trop de fichiers, on risque de saturer le compteur de fichiers ouverts du système d'exploitation.

2. Le fichier de données porte l'extension `csv` qui signifie `Comma Separated Values`. C'est un fichier texte dont chaque ligne est un enregistrement de données sous forme de t-uplet constitué de champs d'informations séparés par le caractère ;. La signification de chaque champ est fixée par la première ligne d'en-têtes. Le fichier peut être édité avec un éditeur de textes comme Notepad++ ou avec un tableur comme LibreOffice Calc.

   Si on choisit d'indexer les champs d'une ligne à partir de 0, quel est l'index du champ *Filières de formations très détaillées*?

3. Recopier et compléter la fonction ci-dessous qui doit retourner le nombre de lignes d'un fichier texte.

```
def nombre_lignes(fichier):
    c = ......
    f = open(fichier)
    for ligne in f:
        c = c..........
    f.close()
    return c
```

4. Quel est le nombre de lignes du fichier `'fr-esr-apb_voeux-et-admissions.csv'` ?

3 Traitement d'une ligne

Méthode *Traitement de ligne*

Lorsqu'on a récupéré une ligne dans un fichier texte, on peut avoir besoin de réaliser les opérations de base suivantes :

☞ La suppression des caractères de fin de ligne avec `rstrip` ou des espaces / tabulations à gauche avec `lstrip`. Les caractères de fin de ligne ne sont pas affichés par défaut mais on peut les visualiser avec l'option adéquate dans un éditeur de textes, ils font référence aux actions des machines électromécaniques qui ont précédé les ordinateurs dans les opérations de traitement de texte : `'\r'` signifie *retour charriot* et `'\n'` *nouvelle ligne*. Les caractères de fin de ligne diffèrent selon les systèmes d'exploitation : `'\r\n'` sous Windows ou `'\n'` sous Linux.

```
In [22]: ligne = 'Un saut de ligne \n'

In [23]: ligne.rstrip()
Out[23]: 'Un saut de ligne'

In [24]: ligne2 = '\t Une tabulation et des espaces à gauche'

In [25]: ligne2.lstrip()
Out[25]: 'Une tabulation et des espaces à gauche'
```

☞ Le découpage de la ligne selon un caractère de séparation pour récupérer les valeurs de différents champs avec la fonction / méthode de chaîne de caractères `split` :

```
In [26]: ligne3 = 'eleve,login,password'

In [27]: ligne3.split(',')
Out[27]: ['eleve', 'login', 'password']
```

L'opération réciproque est possible avec la fonction / méthode de liste `join` :

```
In [28]: champs = ['eleve', 'login', 'password']

In [30]: ';'.join(champs)
Out[30]: 'eleve;login;password'
```

1. Recopier et compléter la fonction `decoupageLigne(ligne, sep)` pour qu'elle retourne la liste des champs constituant une ligne, découpés selon le caractère de séparation passé en paramètre.

```
def decoupageLigne(ligne, sep):
    ligne = ligne.rstrip() #suppression du caractère de fin de ligne
    return ..............
```

2. Quelle instruction permet d'obtenir le nombre de champs dans une ligne du fichier SOURCE ?

3. Recopier et compléter la fonction `listeVoeuxAnnee(fichier, annee, sep)` pour qu'elle retourne, pour l'année passée en paramètre, la liste des lignes d'un fichier structuré comme SOURCE, découpées en listes de champs selon un certain caractère de séparation. On saute la ligne des en-têtes.

```
def listeVoeuxAnnee(fichier, annee, sep):
    voeux = []
    f = open(fichier)
    f.readline() #sauter la ligne des en-tetes
    for ligne in f:
        champs = ................
        if ..................:
            voeux.append(champs)
    f.close()
    return voeux
```

Voici un exemple d'exécution, chaque liste contenue dans la liste de listes retournée contient une ligne (un type de voeu) du fichier SOURCE découpée en champs. Nous appellerons désormais liste de voeux ce type de listes.

```
In [5]: voeux_2016 = listeVoeuxAnnee(SOURCE, '2016', ';')

In [6]: len(voeux_2016)
Out[6]: 9004

Out[8]: voeux_2016[0]
['2016', '0720048L', 'Lycée Raphael ELIZE',...., '6.25', '0']
```

4. Le nombre d'admis est dans le champ d'index 18. Écrire une fonction `nombreAdmis(voeux)` qui prend en paramètres une liste de voeux et qui retourne le nombre total d'admis sur l'ensemble de ces voeux.


Sur certaines lignes, le champ 18 contient la valeur `'inconnu'` dont il ne faut pas tenir compte dans la somme sinon Python va lever une exception.

```
In [10]: nombreAdmis(voeux_2016)
Out[10]: 532747
```

5. Inspecter le fichier SOURCE puis donner un code Python permettant d'obtenir le pourcentage de filles parmi les admis en 2016. Le résultat attendu est `0.5346984591184933`.

4 Filtrage des lignes par champs

1. Recopier et compléter la fonction `filtreUai(voeux, codeUAI)` pour qu'elle retourne, à partir d'une liste de voeux, la sélection des voeux dont le champ 1 a pour valeur le code UAI passé en paramètre.

```
def filtreUai(voeux, codeUAI):
    voeux_filtre = []
    for v in voeux:
        if ..................:
            ...................
    return voeux_filtre
```

Exemple d'exécution :

```
In [14]: voeux_Parc_2016 = filtreUai(voeux_2016, '0690026D')

In [15]: voeux_Parc_2016[0]
Out[15]: ['2016', '0690026D', 'Lycée du Parc', ....., 'Classe pré
    paratoire littéraire', 'Lettres', ...., '83.8709677419']
```

2. Écrire sur le modèle de la précédente, une fonction `filtreCpge(voeux)`.

   Donner un code `Python` permettant d'obtenir le pourcentage de filles admises en CPGE en 2016. Le résultat attendu est `0.46234610802223985`.

3. Écrire une fonction `filtreCpgeDepartement(voeux, departement)` qui retourne, à partir d'une liste de voeux, la sélection des voeux dont le champ 7 a pour valeur `'4_CPGE'` et le champ 3 le numéro du département (en chaîne de caractères).

   Donner un code `Python` permettant d'obtenir le pourcentage d'admis au lycée du Parc parmi les admis en CPGE dans le Rhône en 2016. Le résultat attendu est `0.27071072883657765`.

4. Écrire une fonction `filtre1champ(voeux, indexchamp, valchamp)` qui retourne, à partir d'une liste de voeux, la sélection des voeux dont le champ d'index `indexchamp` a pour valeur `valchamp`.

5. **Question plus difficile**

   Écrire une fonction `filtreNchamps(voeux, liste_indexchamp, liste_valchamp)` qui retourne, à partir d'une liste de voeux, la sélection des voeux dont les champs d'index listés dans la liste `liste_indexchamp` ont les valeurs listées dans la liste `liste_valchamp`.

```
In [14]: filtreNchamps(voeux_2016, [3,7], ['69','4_CPGE']) ==
    filtreCpgeDepartement(voeux_2016, '69')
Out[14]: True
```

5 Tri

Méthode

Pour une liste `t` (les tableaux de Python) d'objets comparables (entiers, caractères ...) , Python propose deux façons de la trier :

- ☞ `sorted(t)` *retourne une nouvelle liste* triée dans l'ordre croissant ;
- ☞ `t.sort()` *trie en place* la liste `t` par ordre croissant.

Dans les deux cas, on peut obtenir un ordre décroissant avec le paramètre optionnel `reverse = True`.
Plus généralement on peut personnaliser le tri avec le paramètre optionnel `key = fonction` : les éléments du tableau seront triés suivant leur image par `fonction`.

Pour comparer des couples (ou `tuple` en Python), par exemple `(x, y)` et `(u, v)`, Python compare d'abord `x` et `u` puis `y` et `v` en cas d'égalité etc ...

```
In [1]: t = ['b','c','a']
In [2]: m = sorted(t)
In [3]: m
```

```
Out[3]: ['a', 'b', 'c']
In [4]: t
Out[4]: ['b', 'c', 'a']
In [5]: t.sort()
In [6]: t
Out[6]: ['a', 'b', 'c']
In [7]: t.sort(reverse=True)
In [8]: t
Out [8]: ['c', 'b', 'a']
In [9]: def clefTri(v):
   ...:     return v[1]
   ...:
In [10]: voeux = [['Parc', 8], ['Fermat', 10]]
In [11]: sorted(voeux)
Out[11]: [['Fermat', 10], ['Parc', 8]]
In [12]: sorted(voeux, key = clefTri)
Out[12]: [['Parc', 8], ['Fermat', 10]]
```

1. On souhaite trier les voeux APB de l'année 2016 par ordre **croissant** du taux de boursier admis (champ d'index 32).

   Expliquer pourquoi on peut utiliser le code Python ci-dessous :

```
voeux_cpge_2016 = filtreCpge(voeux_2016)

def clefTri(v):
    taux_boursier = v[32]
    if taux_boursier == '':
        taux_boursier = 0
    return float(taux_boursier)

tri_boursier_croissant_2016 = sorted(voeux_cpge_2016, key = clefTri)
```

2. Quelle instruction Python permet alors de trier les voeux APB de l'année 2016 par ordre **décroissant** du taux de boursier admis ?

   Pour afficher le classement on pourra utiliser la fonction `affichage_tri(voeux)` fournie dans le fichier `cadeau.py`.

```
In [22]: affichage_tri(tri_boursier_decroissant_2016)

Rang Lycée                          Département Filière                          Taux
1    Lycée Paul Eluard              93          PCSI                             70.5882352941
2    Lycée Baimbridge               971         ECT - Option technologique       65.3846153846
3    Lycée Amiral Pierre Bouvet     974         TSI                              63.6363636364
....
```

3. Écrire un code Python qui permet de trier les voeux APB de l'année 2016 par ordre **décroissant** du taux de boursier admis puis par ordre alphabétique **croissant** de la filière (champ d'index 9).

```
Rang Lycée                          Département Filière                          Taux
1    Lycée Alphonse DAUDET          30          B/L - Lettres et sciences sociales 30.4347826087
2    Lycée Faidherbe                59          B/L - Lettres et sciences sociales 18.1818181818
3    Lycée Albert Chatelet          59          B/L - Lettres et sciences sociales 17.07317
....
```

4. Écrire une fonction `filtreFiliere(voeux, filiere)` qui retourne, à partir d'une liste de voeux de CPGE, la sélection des voeux dont le champ d'index 7 a pour valeur la filière passée en paramètre.

Pour obtenir la liste des voeux MPSI triés dans l'ordre décroissant des taux de boursier admis à partir de la liste `tri_boursier_decroissant_2016` des voeux CPGE 2016 triés par ordre décroissant de taux de boursiers, on écrira :

```
tri_boursier_dec_MPSI = filtreFiliere(tri_boursier_decroissant_2016, 'MPSI')
```

5. Recopier et compléter la fonction `ecrire_fichier(voeux, fichier)` pour qu'elle écrive une liste de voeux dans un fichier texte en ne conservant que quatre champs séparés par le caractère `';'` :

`Rang;Lycée;Département;Filière;Taux de boursiers admis`

Pour reconstituer la ligne on utilisera la méthode `join` des chaînes de caractères, décrite dans la méthode *Traitement de ligne* page 3.

```
def ecrire_fichier(voeux, fichier):
    f = open(fichier,'w')
    f.write('Lycée;Département;Filière;Taux de boursiers admis\n')
    for k in range(len(voeux)):
        ........................
        ........................
    f.close()
```

6. À l'aide de la fonction précédente, créer les fichiers textes correspondants aux classements des voeux par ordre décroissants de taux de boursiers pour les six filières de CPGE du lycée du Parc : `tri_boursiers_ECS - Option scientifique.csv`, `tri_boursiers_Lettres.csv`, `tri_boursiers_PCSI.csv`, `tri_boursiers_MPSI.csv`, `tri_boursiers_BCPST.csv` et `tri_boursiers_B-L - Lettres et sciences sociales.csv`.

Que peut-on remarquer au vu de ces classements ?