On souhaite étudier numériquement une série statistique contenue dans un fichier `donnees.csv`. Chaque patient s'y trouve dans l'ordre de son numéro d'anonymat patient (ici 1, 2, ...)

```
patient,genre,age,taille(cm),poids(kg),etat
1,F,35,167,49,87
2,M,45,176,66,85
⋮
```

Après la suite d'instructions suivante

```
import csv
mfile = open( "./DS2-donnees.csv", "r")
donnees = list( csv.reader(mfile, delimiter=",") )
mfile.close()
```

la variable `donnees` contient un tableau de tableaux.

Ainsi `donnees[1]` renverra le tableau `[ "1", "F", "35", "167", "49", "87" ]`. Ce tableau correspond aux données du patient n° 1. On peut dire que ce patient est un femme de 35 ans mesurant 1,67m, pesant 49kg et étant dans un état de santé que le protocole sanitaire estime à 87%.

Enfin `donnees[1][2]` renverra la valeur d'indice 2 du tableau `donnees[1]`, soit "35".

a) Que renvoie l'instruction `len( donnees[1] )` ? l'instruction `donnees[0]` ? l'instruction `donnees[1][4]` ?

b) Afin de convertir en entier toutes les valeurs (sauf celle d'indice 1) d'un tableau, on a défini la procédure suivante. La corriger.

```
1 def convert(tab: list) -> None
2     tab(0) := int tab(0)
3     for i rang( 2, len[tab] )
4         tab(i) := int tab(i)
```

c) Proposer une procédure `convertDonnees(donnees: list) -> None`, qui convertisse en entier toutes les valeurs du tableau de tableaux `donnees` possibles.

d) À partir de maintenant, on suppose que toutes les valeurs dans le tableau de tableaux `donnees` qui peuvent être converties en entier l'ont été.

Que renvoie la commande `donnees[ donnees[1][0] ]` ? la commande `donnees[ donnees[i][0] ]` ? Que remarquez-vous ?

e) On sait qu'un caractère de la table ASCII est codé sur 1 octet et qu'un entier naturel inférieur à 256 est codé sur 1 octet aussi.

Ainsi le tableau `[ "patient", "genre", "age", "taille(cm)", "poids(kg)", "etat" ]` nécessite 38 octets de stockage mémoire.

Combien faut-il d'octets mémoire pour stocker le tableau `[ 1, "F", 35, 167, 49, 87 ]` ?

f) Supposons que l'on ait 1 Kio de mémoire (1 Kio = 1024 octets), combien de données patients la variable `donnees` peut-elle contenir ?

(tourner la page svp)

g) Afin d'obtenir toutes les valeurs contenues dans une colonnes de données, on propose la fonction suivante :

```
1 def colonneI(donnees: list, i: int) -> list:
2     col = []
3     for j in range(1, len(donnees) ):
4         col += [ donnees[j][i] ]
5     return col
```

Dérouler le code avec $i = 3$ et `donnees` définie ci-dessous :

```
donnees = [ [ "patient", "genre", "age", "taille(cm)", "poids(kg)", "etat" ],
            [         1,     "F",    35,          167,          49,    87  ],
            [         2,     "M",    45,          176,          66,    85  ],
            [         3,     "F",    28,          176,          53,    66  ] ]
```

ligne	donnees	i	col	j	sortie
1	[["patient", ...], ..., [..., 66]]	3			

h) Préciser le calcul de la compléxité temporel de cet algorithme en fonction de $n = len(donnees)$. Cette complexité est-elle logarithmique, linéaire, quasi-linéaire, quadratique, ... ?

i) Quelle instruction faut-il faire pour définir une variables `ages` qui contient tous les âges des patients ?

j) Proposer une fonction `ageMax(tab: list) -> int`, qui renvoie l'âge maximal du tableau `tab` ? Proposer une suite d'instructions `python` qui affiche l'âge maximal des patients.

k) À l'aide de la remarque faite au **d)**, proposer une fonction `agesMax(tab : list) -> list`, qui renvoie les numéros des patients qui ont l'âge maximal.

l) Proposer une fonction `bonneSante(donnees: list, binf: int, bsup: int) -> list`, qui renvoie les numéros des patients qui ont un état de santé entre `binf` et `bsup`.

m) On souhaite former deux groupes de patients : l'un composé de patients en santé précaire (moins de 79%) et l'autre composé de patients en bonne santé (plus de 80%).

L'algorithme consiste à parcourir le tableau de tableaux `donnees` et ajouter dans les groupes des personnes tant que chacun des deux groupes ne contiennent pas au moins 10 femmes et au moins 10 hommes.

On supposera qu'il est possible de faire ces deux groupes à partir de `donnees`. Proposer une fonction `groupes(donnees: list) -> tuple`, qui répond au problème.

i.e. `groupes(donnees)` renverra le tuple de tableaux `( [1,2,...], [3,...] )`.