Chapitre 11. Les données sous forme de table

Table des matières

Remplissez le jupyter notebook suivant en vous aidant de votre livre de Première NSI de Serge BAYS .

- Pour répondre, double-cliquez sur **Réponse** et complétez la zone en-dessous. Puis cliquez sur le bouton *Exécuter*.
- **Important : pour fermer votre jupyter notebook, cliquez sur :**

Fichier / Créer une nouvelle sauvegarde

puis sur :

Fichier / Fermer et Arrêter

- Ecrivez ci-dessous votre prénom et votre nom :

Nom - Prénom :

Chapitre 11. Les données sous forme de table

1. Les dictionnaires

1.1 Définition

Lisez le paragraphe **Définition** en bas de la p. 142

1) Qu'est-ce qu'un dictionnaire ?

Réponse :

2) Quelle est la différence entre un dictionnaire et une liste ?

Réponse :

3) Quels noms portent ces indices ?

Réponse :

4) En synthèse, nous pouvons dire qu'un dictionnaire est ...

Réponse :

5) Que faut-il pour créer un dictionnaire ?

Réponse :

1.2 Construction

Lisez le paragraphe **Construction** du bas de la p. 142 au haut de la p. 144

Il existe divers procédés de construction d'un dictionnaire afin d'éviter d'ajouter les éléments un par un

6) Quelle est la fonction qui permet de convertir une liste de listes à deux éléments en dictionnaire ?

Réponse :

Créer un dictionnaire *d* à partir de la liste *liste* en complétant les instructions ci-dessous puis vérifier que *d* est bien un dictionnaire

In [ ]:
```
liste = [['un', 1],['deux', 2],['trois', 3]]
d = ...
print(d)
```

Un dictionnaire peut aussi être construit par compréhension.

7) Ecrire les instructions, permettant de construire par compréhension le dictionnaire *d1* suivant : {1: 3, 2: 5, 3: 7, 4: 9, 5: 11}.

In [ ]:
```
#  Vos instructions
d1 = {...}
print(d1)
```

8) Ecrire les instructions, permettant de construire par compréhension le dictionnaire *d2* suivant : {'A': 4, 'B': 8, 'C': 12, 'D': 16}.

- Rappel : chr(65) renvoie 'A' ; chr(66) renvoie 'B' etc.

In [ ]:
```
#  Vos instructions
d2 = {...}
print(d2)
```

Dernière méthode, utiliser dict en entrant directement les informations.
par exemple :

d = dict(un=1, deux=2, trois=3)

d contiendra :
{'un': 1, 'deux': 2, 'trois': 3}

- Pour le voir, exécutez le code dans la cellule suivante :

In [ ]:
```
d3 = dict(un=1, deux=2, trois=3)
print(d3)
```

9) Ecrire le programme permettant d'afficher une facture résumée.

En entrée :

- Un dictionnaire avec les couples donnant la correspondance numéro: nom pour tous les produits d'un magasin.

```
    d = {numéro du produit1: nom du produit1, numéro du produit2: nom du pr
oduit2, ...}
```

- Une liste de courses dont les éléments sont des 3-tuples :

```
    liste = [(numéro du produit1, quantité1, prix unitaire1), (numéro du pr
oduit2, quantité2, prix unitaire2) ...]
```

En sortie :

- L'impression d'une facture de la liste de courses ayant cet aspect :

```
    quantité1    nom du produit1       prix unitaire1          prix1
    quantité2    nom du produit2       prix unitaire2          prix2
    ...     ...                     ...                     ...

     total : ...
```

- La facture est une chaîne de caractères composée de plusieurs lignes, chaque ligne correspondant à un article.
- Sur chaque ligne apparaissent dans cet ordre : quantité, nom du produit, prix unitaire, prix total séparés par une tabulation '\t'.
- Les lignes sont séparées avec un saut de ligne '\n'.

- En dernière ligne est affiché le montant total à payer.

Pour afficher un nombre avec deux décimales, vous pourrez utiliser :

```
"%.2f"%qte      (qte étant un flottant)
```

- Pour voir le fonctionnement, exécutez le code suivant :

In [ ]:
```
a = "%.2f"%3.148527

print(a)
```

In [ ]:
```
#  Complétez le programme :
d = {101: '1 L. de lait      ', 102: '20 cL Crème fraîche', 201: 'Bag
uette          ',\
     202: 'Pain tranché      ', 301: 'Crevettes          ', 302: 'Fil
et de Limande  ' }
liste = ((101, 3, 1.01), (102, 2, 2.10), (202, 3, 0.80), (302, 1.5, 1
1.25))

facture = ''  #  C'est une chaine de caractères vide au départ.
total = 0  #  Initialisation du total.

for ligne in liste:
    qte = ligne[1]
    numero = ligne[0]
    nom = d[...]
    prix_unit = ...
    prix_qte = ...
    facture = facture + str("%.2f"%qte) + '\t' + nom + ...
    total = total + ...

print(facture)
print(...)
```

1.3 Utilisation

1.3.1 Accès aux éléments

Lisez le paragraphe **Accès aux éléments** de la p. 144 au milieu de la p. 145

10) Comment n'obtenir que les clés du dictionnaire *d* ?

Réponse :

- Exécutez la ligne de code suivante :

In [ ]:
```
d.keys()
```

11) Comment n'obtenir que les valeurs du dictionnaire *d* ?

Réponse :

- Exécutez la ligne de code suivante :

In [ ]:
```
d.values()
```

12) Comment obtenir l'ensemble des couples du dictionnaire *d* ?

Réponse :

- Exécutez la ligne de code suivante :

In [ ]:
```
d.items()
```

13) Que permet le mot clé *in* ? Que ne permet-il pas ?

Réponse :

- Exécutez la ligne de code suivante :

In [ ]:
```
cle_appartient = 102 in d
valeur_appartient = '20 cL Crème fraîche' in d

print(cle_appartient)
print(valeur_appartient)
```

14) Lorsque nous itérons avec une boucle *for* sur un dictionnaire, sur quoi porte la variable d'itération ?

Réponse :

Les questions de 15 à 18 portent sur le dictionnaire *dd*.
Exécutez la cellule ci-dessous pour que ce jupyter notebook mémorise la valeur de la variable *dd*.

In [ ]:
```
dd = {101: '1 L. de lait', 102: '20 cL Crème fraîche', 201: 'baguette
', 202:'pain tranché', 301: 'Crevettes', 302: 'Filet de Limande' }
```

15) Comment vérifier que 201 est une clé et que 303 n'en est pas une ? Ecrire les instructions.

In [ ]:
```
# Réponse pour 201
```

In [ ]:
```
#  Réponse pour 303
```

16) dd.keys() est un objet du type *dict_keys*.

Construire une liste **dd_cles** des clés de dd en itérant sur *dd.keys()* par une boucle for.

In [ ]:
```
#  Réponse :

dd_cles = []
for cle in dd.keys():
    dd_cles.append(...)

print(dd_cles)
```

17) dd.values() est un objet du type *dict_values*.
Construire une liste **dd_valeurs** des valeurs de dd en itérant sur *dd.values()* par une boucle for.

In [ ]:
```
# Réponse :

dd_valeurs = []
for valeur in dd.values():
    dd_valeurs.append(...)

print(...)
```

18) dd.items() est un objet du type *dict_items*.
Construire une liste **dd_couples** des couples de dd en itérant sur *dd.items()* par une boucle for.

In [ ]:
```
# Réponse :

dd_couples = []
for couple in ...():
    dd_couples.append(...)

print(...)
```

1.3.2 Nombre d'éléments

Lisez le paragraphe **Nombre d'éléments** du milieu de la p. 145 au haut de la p. 146

19) Quand je saisis len(dd). que vais-je obtenir ?

Réponse :

20) Qu'est-ce que la longueur d'un dictionnaire ?

Réponse :

21) Dans le programme ci-dessous, une erreur est renvoyée. Pourquoi ?

In [ ]:
```
liste = ['un', 'deux', 'troas', 'quatre']
liste[4] = 'cinq'
```

Réponse :

22) Dans le programme ci-dessous, une erreur est-elle renvoyée ? Pourquoi ?

In [ ]:
```
liste = ['un', 'deux', 'troas', 'quatre']
liste[2] = 'trois'
```

In [ ]:
```
print(liste)
```

Réponse :

23) Dans le programme ci-dessous, une erreur est-elle renvoyée. Pourquoi ?

In [ ]:
```
dico = {1: 'un', 2: 'deux', 3: 'troas', 4: 'quatre'}
dico[5]='cinq'
```

In [ ]:
```
print(dico)
```

Réponse :

24) Que ce passe-t-il si j'exécute l'instruction : dico[2] = 'trois' ?

In [ ]:
```
dico = {1: 'un', 2: 'deux', 3: 'troas', 4: 'quatre'}
dico[2]='trois'
print(dico)
```

Réponse :

Il faut bien faire attention à la clé, dans notre exemple, placée entre les crochets.

Pour *liste* le 2 correspond à la troisième valeur, c'est à dire 'troas'.
Pour *dico* le 2 correspond à la clé dont la valeur est 2, et c'est pour cela qu'il remplace 'deux' par 'trois'
Il faut bien faire la différence entre un indice et une clé.

Il faut noter que la clé n'est pas forcément un nombre dans un dictionnaire, on peut avoir dico['2'].
Regardez ce qui va se passer dans ce cas pour le dictionnaire *dico.*

In [ ]:
```
dico = {1: 'un', 2: 'deux', 3: 'troas', 4: 'quatre'}
dico['2']='deux'
print(dico)
```

Il a ajouté une clé non numérique '2' associée à la valeur 'deux'.
Attention donc aux clés que vous gérez, pour avoir une cohérence dans votre dictionnaire (tout dépend donc comment vous le définissez).

1.3.3 Fonctions, opérations, méthodes

Lisez le paragraphe **Fonctions, opérations, méthodes** du haut de la p. 146 au bas de la p. 147

On considère deux catalogues présentés sous la forme de dictionnaires :

In [ ]:
```
dico1 ={'A001': 'Tabouret Elian', 'A010': 'Table extérieure Woody', '
B002': 'Visseuse Bosch',\
        'B004': 'Visseuse Parkside', 'B007': 'Ponceuse Redstone', 'B0
10': 'Perceuse B&D' }
dico2 ={'A002': 'Tabouret Woody', 'A010': 'Table extérieure Woody', '
B003': 'Visseuse B&D',\
        'B004': 'Visseuse Parkside', 'B008': 'Ponceuse Bosch', 'B011'
: 'Perceuse Makita' }
```

25) Quelle(s) instruction(s) faut-il saisir pour lister les codes de référence se trouvant dans les deux catalogues ?

In [ ]:
```
#  Réponse :
#  Les codes de référence sont les clés. Pour lister les clés commune
s, on utilise d1.keys() & d2.keys()
#  Le résultat renvoyé est l'ensemble des clés figurant dans les deux
dictionnaires simultanément.

print(...&...)
```

26) Quelle(s) instruction(s) faut-il saisir pour lister les codes de référence et les noms des articles se trouvant dans les deux catalogues (présentés sous forme de couples) ?

In [ ]:
```
#  Réponse :
#  Les codes de référence et les noms des articles sont les items. Po
ur lister les items communs,
#  on utilise  d1.items() & d2.items()
#  Le résultat renvoyé est l'ensemble des items figurant dans les deu
x dictionnaires simultanément.

print(...&...)
```

27) Quelle(s) instruction(s) faut-il saisir avoir les codes de référence des produits qui sont dans dico1 mais pas dans dico2 ?

In [ ]:
```
#  Réponse :
#  Les codes de référence sont les clés. Pour lister les clés qui son
t présentes dans dico1 et qui
#  sont absentes dans dico2, on utilise d1.keys() - d2.keys()
#  Le résultat renvoyé est l'ensemble des clés figurant dans dico1 ma
is pas dans dico2.

print(... - ...)
```

28) Quelle est l'instruction permettant d'effacer un dictionnaire *dico* ?

In [ ]:
```
dico = {1: 'un', 2: 'deux', 3: 'troas', 4: 'quatre'}
print(dico)
dico....
print(dico)
```

Réponse :

29) Quelle instruction saisir pour effacer, dans *dico1* l'élément dont la clé est 'B007' ?

In [ ]:
```
dico1 ={'A001': 'Tabouret Elian', 'A010': 'Table extérieure Woody', '
B002': 'Visseuse Bosch',\
        'B004': 'Visseuse Parkside', 'B007': 'Ponceuse Redstone', 'B0
10': 'Perceuse B&D' }
print(dico1)
del(...)
print(dico1)
```

Réponse :

30) Que se passe-t-il si je saisis l'instruction print(dico1['A002']) ?

In [ ]:
```
print(dico1[...])
```

Réponse :

31) Comment faire pour éviter cela ?

Réponse : On utilise la méthode ...

32) Utiliser cette méthode pour corriger l'erreur obtenue à la question 30.

In [ ]:
```
print(dico1.get(...))
```

33) Que va retourner Python lorsque j'exécute l'instruction de la question 32 ?

Réponse :

1.3.4 Copie

Lisez le paragraphe **Copie** du bas de la p. 147 au bas de la p. 149

Pour la copie, les comportements sont similaires à ceux rencontrés avec les listes.

34) Si je saisis et exécute les instructions suivantes

```
dico3 ={1: 'Chat', 2: 'Chien', 3: 'Cheval', 4: 'Lion', 5: 'Panda', 6: 'Vipè
re' }
dico4 = dico3

print(dico3)
print(dico4)
print()

dico4[1]= 'Abeille'

print(dico3)
print(dico4)
```

que se passe-t-il ?

In [ ]:

Réponse :

35) Quelle instruction saisir pour éviter d'avoir ce problème ?

In [ ]:
```
dico3 ={1: 'Chat', 2: 'Chien', 3: 'Cheval', 4: 'Lion', 5: 'Panda', 6:
'Vipère' }
dico4 = ...

print(dico3)
print(dico4)
print()

dico4[1]= 'Abeille'

print(dico3)
print(dico4)
```

Réponse :

36) Quelle est l'autre instruction possible ?

In [ ]:
```
dico3 ={1: 'Chat', 2: 'Chien', 3: 'Cheval', 4: 'Lion', 5: 'Panda', 6:
'Vipère' }
dico4 = ...

print(dico3)
print(dico4)
print()

dico4[1]= 'Abeille'

print(dico3)
print(dico4)
```

Réponse :

La méthode dictionnaire.copy() vue avant concerne des dictionnaires dont les valeurs sont de type simple. Nous allons voir comment faire pour des dictionnaires dont les valeurs sont des listes.

37) On utilise la méthode copy pour un dictionnaire de listes. Que se passe-t-il pour dico5 ?

In [ ]:
```
dico5 ={'Paul': [10, 12, 12], 'Marc': [8, 14, 18], 'Jean': [12, 7, 15
]}
dico6 = dico5.copy()

print(dico5)
print(dico6)
print()

dico6['Paul'][2] = 20

print(dico5)
print(dico6)
```

Réponse :

38) Quelle instruction faut-il utiliser pour copier un dictionnaire de listes, lorsqu'on veut pouvoir modifier la copie tout en gardant intact l'original ?

Réponse :

39) Corriger les instructions ci-dessous pour que ce problème n'arrive pas.

In [ ]:
```
#  Votre réponse :

from copy import deepcopy
dico5 ={'Paul': [10, 12, 12], 'Marc': [8, 14, 18], 'Jean': [12, 7, 15
]}
dico6 = ...

print(dico5)
print(dico6)
print()

dico6['Paul'][2] = 20

print(dico5)
print(dico6)
```

- Tout comme les éléments d'une liste peuvent être des listes, les valeurs des dictionnaires peuvent être des dictionnaires.

Création d'un dictionnaire par la méthode fromkeys :

In [ ]:
```
{}.fromkeys([liste de clés], valeur_partout_la_meme)
#  ou encore :
dict.fromkeys([liste de clés], valeur_partout_la_meme)
```

- Pour créer un nouveau dictionnaire en donnant une liste de clés et **une même valeur pour tous les items**.
- Exécutez le code suivant :

- Exécutez le code suivant :

In [ ]:
```
nouveau_dico1 = {}.fromkeys([1, 2, 3, 4], 'inconnu')

print("nouveau_dico1 = ", nouveau_dico1)
```

- La même chose mais en écrivant dict.fromkeys au lieu de {}.fromkeys
- Exécutez le code suivant :

In [ ]:
```
nouveau_dico2 = dict.fromkeys([1, 2, 3, 4, 5, 6], 0.365)

print("nouveau_dico2 = ", nouveau_dico2)
```

Créer un dictionnaire de couples clé - valeur dont les valeurs sont des dictionnaires

- Création d'un dictionnaire où toutes les valeurs sont des dictionnaires écrits manuellement.
- Exécutez le code suivant :

In [ ]:
```
dico_de_dicos1 = {1: {'Nom': 'Duvergnat', 'Année_de_naissance': 1964}
,\
                  2: {'Nom': 'Blondel', 'Année_de_naissance': 1966},\
                  3: {'Nom': 'Brard', 'Année_de_naissance': 1993}}
print("dico_de_dicos1 = ", dico_de_dicos1)
```

- La même chose, mais l'une des valeurs est un dictionnaire créé avec la méthode .fromkeys
- Exécutez le code suivant :

In [ ]:
```
#  Création d'un dictionnaire de dictionnaires avec l'un des dictionn
aires créés à l'aide de la méthode fromkeys

dico_de_dicos2 = {1: {'Nom': 'Duvergnat', 'Année_de_naissance': 1964}
,\
                  2: {'Nom': 'Blondel', 'Année_de_naissance': 1966},\
                  3: {}.fromkeys(['Nom', 'Année_de_naissance'], 'cham
```

```
p_vide')}
print("dico_de_dicos2 = ", dico_de_dicos2)
```

En combinant ces deux notions :

- Le troisième dictionnaire de cet exemple est un dictionnaire ".fromkeys"
- Exécutez le code suivant :

In [ ]:
```
pays = {'France': {'capitale': 'Paris' , 'population': 64897954, 'sup
erficie': 643800.0},
        'Portugal': {'capitale': 'Lisbonne' , 'population': 10300000,
'superficie': 92300.0},
        'nouveau': {}.fromkeys(['capitale', 'population', 'superficie
'], 'inconnu')}

print("pays = ", pays)
```

Extraction d'une valeur par sa clé

- Exécutez les codes suivants :

In [ ]:
```
print(pays.get('France'))
```

In [ ]:
```
print(pays.get('nouveau'))
```

In [ ]:
```
print(pays.get('Allemagne'))
```

40) Que se passe-t-il pour la dernière instruction ? Et pourquoi ?

Réponse :

41) Ajouter dans le dictionnaire les informations concernant l'Allemagne : Berlin, 82073100

41) Ajouter dans le dictionnaire les informations concernant l'Allemagne : Berlin, 83073100 , 357386.0. Quelle instruction allez-vous utiliser ?

In [ ]:
```
pays['Allemagne'] = ...
```

In [ ]:
```
print(pays.get('Allemagne'))
```

42) Quelle instruction allez-vous utiliser pour modifier la population Française (en y ajoutant les DOM) soit 67063703 (source INSEE)?

In [ ]:
```
pays['France']['population'] = ...
```

In [ ]:
```
print(pays.get('France'))
```

43) Pourquoi est-il préférable, dans certains cas, d'utiliser un dictionnaire plutôt qu'une liste de listes ?

Réponse :

- De plus, dans un dictionnaire, on peut appeler un élément par une clé qui n'existe pas, sans provoquer d'erreur, du moment qu'on utilise mon_dictionnaire.get(key)

1.4 Application

Lisez le paragraphe **Application** p. 150 et en haut de la p. 151

Voici quelques instructions qui permettent de récupérer les informations exif de l'image image_2.jpg

In [ ]:
```
from PIL import Image # PIL est la bibliothèque spécialisée dans le t
raitement de l'image
```

```
import requests # Cette bibliothèque permet de visiter une adresse in
ternet
from io import BytesIO # module pour transformer en bytes

image = requests.get('http://www.famille-reboul.fr/NSI/seq_11/image_2
.jpg')
#  img = PIL.Image.open()
imgpil = Image.open(BytesIO(image.content))
exif_data = imgpil._getexif()
exif_data
```

44) a) Où a été prise cette photo ? (aidez vous de ce site : https://www.coordonnees-gps.fr/)

Réponse :

44) b) Avec quel appareil ?

Réponse :

44) c) Quelles sont les dimensions de cette photo ?

Réponse :

44) d) Quelle est la résolution en ppp ?

Réponse :

44) e) Quel jour et à quelle heure a-t-elle été prise ?

Réponse :

2. Traitement de données en tables

2.1 Introduction

Lisez le paragraphe **Traitement de données en tables** en bas de la p.151

45) Comment étaient stockées les informations avant 1970 ?

Réponse :

46) Qui a émis le premier la théorie du modèle relationnel ? (voir aussi dans les vidéos)

Réponse :

47) Que signifie SGBD ?

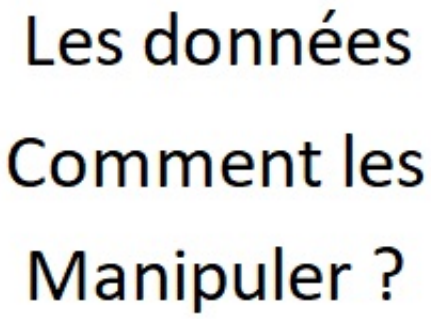
Réponse :

48) Que permet un SGBD ?

Réponse :

Deux vidéos explicatives à regarder (cliquer sur les titres)

(http://www.famille-reboul.fr/NSI/seq_11/histoire_des_bases_de_donnees.mp4)



(http://www.famille-reboul.fr/NSI/seq_11/Donnees_comment_les_manipuler.mp4)

2.2 Importation d'une table

Lisez le paragraphe **Importation d'une table** p. 152 et haut de la p. 153

Nous allons récupérer un fichier csv. Les données sont séparées par des points virgules ou des virgules (ce qui sera la cas pour notre fichier). Ce fichier sera enregistré dans une liste.

Télécharger le fichier Departements.csv ici et mettez le dans le même répertoire que le fichier jupyter (http://www.famille-reboul.fr/NSI/seq_11/Departements.csv)

- Par un clic droit sur Departements.csv, ouvrez-le avec Text Editor pour y jeter un coup d'oeil.
- On constate que le symbole de séparation utilisé dans ce fichier csv est la virgule.
  *D'ailleurs, **csv** est l'acronyme de Comma-Separated Values*

- Fermez Text Editor.
- Cliquez sur la Framboise / Bureautique / LibreOffice Calc.
- Dans le tableur Calc, cliquez sur File / Open et trouvez le fichier Departements.csv
- Dans la boite de dialogue qui s'ouvre, sélectionnez comme séparateur seulement "Comma" (c'est à dire virgule).

49) Quelle est donc la première opération effectuée par LibreOffice Calc en ouvrant ce fichier ?

Réponse :

50) Quel sont les domaines des valeurs des champs du fichier que l'on va importer ?

Réponse :

Champs (ou attributs)

Enregistrements

Code	Nom	Cantons	Communes	Population	Region	Chef_Lieu	Superficie (km²)
1	Ain	23	393	659180	Auvergne-Rhône-Alpes	Bourg-en-Bresse	5762
2	Aisne	21	800	546527	Hauts-de-France	Laon	7369
3	Allier	19	317	347035	Auvergne-Rhône-Alpes	Moulins	7340
4	Alpes-de-Haute-Provence	15	198	168381	Provence-Alpes-Côte d'Azur	Digne	6925
5	Hautes-Alpes	15	162	145883	Provence-Alpes-Côte d'Azur	Gap	5549
6	Alpes-Maritimes	27	163	1097496	Provence-Alpes-Côte d'Azur	Nice	4299
7	Ardèche	17	335	334688	Auvergne-Rhône-Alpes	Privas	5529
8	Ardennes	19	449	280032	Grand Est	Charleville-Mézières	5229
9	Ariège	13	327	157210	Occitanie	Foix	4890
10	Aube	17	431	317118	Grand Est	Troyes	6004
11	Aude	19	433	379094	Occitanie	Carcassonne	6139
12	Aveyron	23	285	289488	Occitanie	Rodez	8735
13	Bouches-du-Rhône	29	119	2048504	Provence-Alpes-Côte d'Azur	Marseille	5088
14	Calvados	25	527	708344	Normandie	Caen	5548
15	Cantal	15	246	150185	Auvergne-Rhône-Alpes	Aurillac	5726
16	Charente	19	366	361539	Nouvelle-Aquitaine	Angoulême	5956
17	Charente-Maritime	27	463	659968	Nouvelle-Aquitaine	La Rochelle	6864
18	Cher	19	287	311456	Centre-Val de Loire	Bourges	7235
19	Corrèze	19	280	249135	Nouvelle-Aquitaine	Tulle	5857
2A	Corse-du-Sud	11	124	159768	Corse	Ajaccio	4014
2B	Haute-Corse	15	236	180465	Corse	Bastia	4666
21	Côte-d'Or	23	700	545798	Bourgogne-Franche-Comté	Dijon	8763
22	Côtes-d'Armor	27	348	617107	Bretagne	Saint-Brieuc	6878
23	Creuse	15	256	122133	Nouvelle-Aquitaine	Guéret	5565

51) Compléter le programme ci-dessous pour que les champs se trouvent dans *champs* et les valeurs dans *lignes*

Il faudra, de plus, convertir les champs 'cantons', 'commune', 'population', et 'Superficie' en entiers.

Pour vérifier que l'import s'est bien fait, vous pourrez afficher *champs2* et *table*

In [ ]:
```
#  Pour importer une table avec Modification de type pour les champs
contenant des nombres entiers
f = open(..., ...)
champs = f. ...() #  Lecture de la première ligne
lignes = f. ...() #  Lecture des autres lignes
table=[]

champs2 = champs.rstrip().split(',') # Met les champs dans une liste

for ligne in lignes:
    liste = ligne.rstrip(). ...(...)
    liste[...] = int(liste[...])  # Modification de domaine de valeur
pour transformer une chaine de caractères en entier
    liste[...] = int(liste[...])
    liste[...] = int(liste[...])
    liste[...] = int(liste[...])
    table.append(liste)
f.close()

print("champs2 = ", champs2)
print()
print("table = ", table)
```

2.3 Recherche dans une table

Lisez le paragraphe **Recherche dans une table** du milieu de la p. 153 au bas de la p. 154

On veut rechercher tous les départements dont la population est supérieure à 500 000 habitants.

52) Compléter le programme permettant de créer une liste contenant les noms de ces départements

In [ ]:
```
#  La liste champs2 a été fabriquée à la question 51.
#  Recherche des indices des champs nom et population dans la liste c
hamps2.
indice_nom = champs2.index('Nom')
indice_pop = champs2.index('Population')
```

```
# A vous de compléter
rep = []
for ligne in table:  #  table a été fabriquée à la question 51.
    if ...
        ...
print(rep)
```

- On veut regrouper les départements par régions et créer une liste contenant des listes de départements rangés par région.

Exemple :

```
[[...], ['Côtes d'Armor', 'Ile et Villaine', 'Morbihan', 'Finistère'], ...,
[...]]
```


53) Complétez ce programme

In [ ]:

```
#  La liste champs2 a été fabriquée à la question 51.
#  Recherche des indices des champs nom et population dans la liste c
hamps2.
indice_region = champs2.index('Region')
indice_nom = champs2.index('Nom')

liste_regions = ['Auvergne-Rhône-Alpes', 'Bourgogne-Franche-Comté', '
Bretagne', 'Centre-Val de Loire',\
                 'Corse', 'Grand Est', 'Guadeloupe', 'Guyane', 'Hauts
-de-France', 'Ile-de-France',\
                 'La Réunion', 'Martinique', 'Mayotte', 'Normandie',
'Nouvelle-Aquitaine', 'Occitanie',\
                 'Pays de la Loire', "Provence-Alpes-Côte d'Azur"]

#  Création de list_regroup qui est la liste contenant les départemen
ts regroupés par région.
#  Ce sera une liste de listes :
#  Liste_regoup[0] contiendra tous les noms des départements de la ré
gion Auvergne-Rhône-Alpes
#  Liste_regoup[1] contiendra tous les noms des départements de la ré
gion Bourgogne-Franche-Comté
#  Liste_regoup[2] contiendra tous les noms des départements de la ré
gion Bretagne
```

```
#  Liste_regoup[3] contiendra tous les noms des départements de la ré
gion Centre-Val de Loire
#  Liste_regoup[4] contiendra tous les noms des départements de la ré
gion Corse
#  Liste_regoup[5] contiendra tous les noms des départements de la ré
gion Grand Est
#  Liste_regoup[6] contiendra tous les noms des départements de la ré
gion Guadeloupe
#  Liste_regoup[7] contiendra tous les noms des départements de la ré
gion Guyane
#  Liste_regoup[8] contiendra tous les noms des départements de la ré
gion Hauts-de-France
#  Liste_regoup[9] contiendra tous les noms des départements de la ré
gion Ile-de-France
#  Liste_regoup[10] contiendra tous les noms des départements de la r
égion La Réunion
#  Liste_regoup[11] contiendra tous les noms des départements de la r
égion Martinique
#  Liste_regoup[12] contiendra tous les noms des départements de la r
égion Mayotte
#  Liste_regoup[13] contiendra tous les noms des départements de la r
égion Normandie
#  Liste_regoup[14] contiendra tous les noms des départements de la r
égion Nouvelle-Aquitaine
#  Liste_regoup[15] contiendra tous les noms des départements de la r
égion Occitanie
#  Liste_regoup[16] contiendra tous les noms des départements de la r
égion Pays de la Loire
#  Liste_regoup[17] contiendra tous les noms des départements de la r
égion Provence-Alpes-Côte d'Azur

liste_regroup = [[] for i in range(...)] #  Création de la liste de l
istes. Il y a 18 régions.

for ligne in table :  #  table a été fabriquée à la question 51.
    # Exemple : ligne = ['1', 'Ain', 23, 393, 659180, 'Auvergne-Rhône
-Alpes', 'Bourg-en-Bresse', 5762]

    index_reg = liste_regions.index(ligne[...])
    #  Exemple : index_reg = liste_regions.index('Auvergne-Rhône-Alpe
s')

    liste_regroup[index_reg].append(ligne[...])
    #  Exemple : liste_regroup[0].append('Ain')
```

In [ ]:

```
print(liste_regroup)
```

2.4 Tri d'une table

Lisez le paragraphe **Tri d'une table** du bas de la p. 154 au milieu de la p.155

On veut trier les lignes de *table* dans l'ordre croissant de la densité de population en nombre d'habitants par km². Une liste sera créée contenant, le nom du département, la population, la superficie et la densité.

54) Ecrire le programme correspondant

In [ ]:

```
#  Recherche de l'indice des champs population, nom et 'Superficie (km²)'
indice_superf = champs2.index('Superficie (km²)')  #  Indice du champ 'Superficie (km²)' dans champs2
indice_nom = champs2.index('Nom')                  #  Indice sur champ 'Nom' dans champs2
indice_popu = champs2.index('Population')          #  Indice sur champ 'Population' dans champs2


def densite(enregistrement):
    """
    Renvoie la densité de population dans l'enregistrement

    Parametres nommes
    -----------------
    enregistrement : de type list
                     C'est la liste des données d'un département

    Retourne
    --------
    densite : de type float
            C'est le quotient de la division Population / Superficie (km²)

    """

    densite = enregistrement[indice_popu] / enregistrement[indice_superf]
    return densite

# A vous de continuer. Indication : vous pouvez reproduire la méthode vue à la question 39) de la
# séquence 9 Algorithmes de tri et algorithmes gloutons.

#  Tri de la table 'table' selon le critère densité
#  'table' a 8 champs : 'Code', 'Nom', 'Cantons', 'Communes', 'Popula
```

```
#  table à 8 champs : 'Code', 'Nom', 'Cantons', 'Communes', 'Popula
tion', 'Region', 'Chef_Lieu', 'Superficie'
tri = sorted(..., key=..., reverse=...)
print("tri = ", tri)
print()



#  Création d'une table rep qui contient uniquement les champs Nom, P
opulation, Superficie, Densité
rep = []
for elt in tri:
    ma_densite = elt[...]/elt[...]
    rep.append([elt[...], elt[...], elt[...], "%.2f"%ma_densite])
print("rep = ", rep)
```

55) Ecrire le programme permettant de trouver la place de la Mayenne dans ce classement

In [ ]:
```
#  Votre programme
...
...
...
...

print("La Mayenne est au rang ", rang)
```

3. Fusion de tables (concaténation ou jointure)

Lisez le **paragraphe d'introduction (2 lignes)** au milieu de la p. 155

56) Quelle est la différence entre une concaténation de tables et une jointure de tables ?

Réponse :

- Concaténation : ...
- Jointure : ...

Concaténation : exemple avec deux tables l'une en dessous de l'autre ayant les mêmes colonnes

Nom	Continent	Superficie	Population	Capitale
Afghanistan	Asie	652864	34124800	Kaboul
Angola	Afrique	1246700	30355900	Luanda
Albanie	Europe	28748	3048000	Tirana
Andorre	Europe	468	85600	Andore-La-Vieille
Argentine	Amérique du sud	2791800	44293300	Buenos Aires
Armenie	Asie	29800	3045200	Erevan
Australie	Oceanie	7692060	23470100	Canberra
Autriche	Europe	83870	8754400	Vienne

Nom	Continent	Superficie	Population	Capitale
Afrique du Sud	Afrique	1219910	58775020	Johannesbourg
Algérie	Afrique	2391740	42972880	Alger
Allemagne	Europe	357390	83149300	Berlin
Arabie saoudite	Asie	2149690	34173500	Riyad

La concaténation donne :

Nom	Continent	Superficie	Population	Capitale
Afghanistan	Asie	652864.0	34124800	Kaboul
Angola	Afrique	1246700.0	30355900	Luanda
Albanie	Europe	28748.0	3048000	Tirana
Andorre	Europe	468.0	85600	Andore-La-Vieille
Argentine	Amérique du sud	2791800.0	44293300	Buenos Aires
Armenie	Asie	29800.0	3045200	Erevan
Australie	Oceanie	7692060.0	23470100	Canberra
Autriche	Europe	83870.0	8754400	Vienne
Afrique du Sud	Afrique	1219910.0	58775020	Johannesbourg
Algérie	Afrique	2391740.0	42972880	Alger
Allemagne	Europe	357390.0	83149300	Berlin
Arabie saoudite	Asie	2149690.0	34173500	Riyad

Jointure : exemple avec deux tables utilisant un champ commun (dans cet exemple c'est le champ 'Nom')

Nom	Continent	Superficie	Population	Capitale
Afghanistan	Asie	652864	34124800	Kaboul
Angola	Afrique	1246700	30355900	Luanda
Albanie	Europe	28748	3048000	Tirana
Andorre	Europe	468	85600	Andore-La-Vieille
Argentine	Amérique du sud	2791800	44293300	Buenos Aires
Armenie	Asie	29800	3045200	Erevan
Australie	Oceanie	7692060	23470100	Canberra
Autriche	Europe	83870	8754400	Vienne

Nom	PIB (millards $/an)	Langue officielle
Afghanistan	19.59	Dari et Pachto
Angola	107.32	Portugais
Albanie	15.2	Albanais
Andorre	3.01	Catalan
Argentine	519.0	Espagnol, Guarani, Toba, Mocovi, Wichi
Armenie	12.41	Arménien
Australie	1432.0	Anglais
Autriche	456.0	Allemand

La jointure donne :

Nom	Continent	Superficie	Population	Capitale	PIB (millards $/an)	Langue officielle
Afghanistan	Asie	652864.0	34124800	Kaboul	19.59	Dari et Pachto
Angola	Afrique	1246700.0	30355900	Luanda	107.32	Portugais
Albanie	Europe	28748.0	3048000	Tirana	15.2	Albanais
Andorre	Europe	468.0	85600	Andore-La-Vieille	3.01	Catalan
Argentine	Amérique du sud	2791800.0	44293300	Buenos Aires	519.0	Espagnol, Guarani, Toba, Mocovi, Wichi
Armenie	Asie	29800.0	3045200	Erevan	12.41	Arménien
Australie	Oceanie	7692060.0	23470100	Canberra	1432.0	Anglais
Autriche	Europe	83870.0	8754400	Vienne	456.0	Allemand

3.1 Concaténation de tables

Lisez le paragraphe **Concaténation de tables** au milieu de la p. 155 au haut de la p. 156

57) Quel point d'attention faut-il quand on concatène des tables ?

Réponse :

Télécharger les deux fichiers *region2019_dom.csv* et *metropole2019_dom.csv* suivants et les enregistrer dans le dossier contenant ce fichier jupyter.
Table des régions des DOM (http://www.famille-reboul.fr/NSI/seq_11/region2019_dom.csv)
Table des régions de la métropole (http://www.famille-reboul.fr/NSI/seq_11/region2019_metropole.csv)

reg	cheflieu	tncc	ncc
1	97105	3	GUADELOUPE
2	97209	3	MARTINIQUE
3	97302	3	GUYANE
4	97411	0	LA REUNION
6	97608	0	MAYOTTE

region2019_dom.csv

reg	cheflieu	tncc	ncc
11	75056	1	ILE DE FRANCE
24	45234	2	CENTRE VAL DE LOIRE
27	21231	0	BOURGOGNE FRANCHE COMTE
28	76540	0	NORMANDIE
32	59350	4	HAUTS DE FRANCE
44	67482	2	GRAND EST
52	44109	4	PAYS DE LA LOIRE
53	35238	0	BRETAGNE
75	33063	3	NOUVELLE AQUITAINE
76	31555	1	OCCITANIE
84	69123	1	AUVERGNE RHONE ALPES
93	13055	0	PROVENCE ALPES COTE D AZUR
94	2A004	0	CORSE

region2019_metropole.csv

58) Concaténer ces deux tables pour créer une table *liste_regions* (Les champs ont les mêmes noms, sont dans le même ordre et ont les mêmes domaines de valeurs).
liste_regions doit être une liste de listes respectant les deux conditions :

- Ne contenir que des chaînes de caractères.
- Avoir en première ligne la liste des champs.

In [ ]:

```
# Ecrire ci-dessous les instructions permettant de concaténer les deux fichiers pour
# créer la table liste_regions (qui est une liste de listes)

#  Importation de la table des régions DOM
#  Pour importer une table avec Modification de type pour les champs contenant des nombres entiers
f = open('region2019_dom.csv','r')
champs = ... #  Lecture de la première ligne
lignes = ...  #  Lecture des autres lignes
table_dom = []

champs2 = champs.rstrip().split(',') # Met les champs dans une liste

for ligne in lignes:
    liste = ligne.rstrip().split(",")
    liste[0] = int(liste[0])  # Modification de domaine de valeur pour transformer une chaine de caractères en entier
    liste[2] = int(liste[2])
    table_dom.append(liste)
f.close()

print("champs2 = ", champs2)
print("table_dom = ", table_dom)
print()


#  Importation de la table des régions Métropole
#  Pour importer une table avec Modification de type pour les champs contenant des nombres entiers
f = open('region2019_metropole.csv','r')
champs = ... #  Lecture de la première ligne
lignes = ...  #  Lecture des autres lignes
table_metropole=[]

champs2 = ... # Met les champs dans une liste

for ligne in lignes:
    liste = ligne.rstrip().split(",")
    table_metropole.append(liste)
f.close()

print("champs2 = ", champs2)
print("table_metropole = ", table_metropole)
print()


#  Concaténation des tables "table_champs2", table_dom" et "table_metropole"
```

```
table_champs2 = [champs2]
liste_region = ...
print("liste_region = ", liste_region)
```

Pour information : il est possible de convertir une liste de listes en un fichier csv

- Dans ce qui précède, on a "converti" un fichier texte au format csv en une liste de listes. Si on a une liste de listes de chaînes de caractères, alors il est possible de faire la conversion dans l'autre sens en important la bibliothèque **csv** et en utilisant une syntaxe un peu particulière qui est illustrée dans l'exemple ci-dessous :

In [ ]:
```
#  Script qui utilise la bibliothèque csv pour transformer une liste de listes en un fichier csv.

import csv

#  Ouverture du fichier avec 'with' ce qui évite de gérer la fermeture.
with open('liste_region.csv', 'w', newline='') as f:  #  newline='' évite des sauts de lignes supplémentaires.
    writer = csv.writer(f, delimiter=',', quotechar='|', quoting=csv.QUOTE_MINIMAL)  #  Fabrication du writer.
    writer.writerows(liste_region)  #  writerows écrit en une seule fois toute la liste de listes.
```

- La liste de listes **liste_region** obtenue à la question 58, qui a été transformée en fichier *liste_region.csv*, est ainsi visible dans un tableur :

reg	cheflieu	tncc	ncc
1	97105	3	GUADELOUPE
2	97209	3	MARTINIQUE
3	97302	3	GUYANE
4	97411	0	LA REUNION
6	97608	0	MAYOTTE
11	75056	1	ILE DE FRANCE
24	45234	2	CENTRE VAL DE LOIRE
27	21231	0	BOURGOGNE FRANCHE COMTE
28	76540	0	NORMANDIE
32	59350	4	HAUTS DE FRANCE
44	67482	2	GRAND EST
52	44109	4	PAYS DE LA LOIRE
53	35238	0	BRETAGNE
75	33063	3	NOUVELLE AQUITAINE
76	31555	1	OCCITANIE

84	69123	1	AUVERGNE RHONE ALPES
93	13055	0	PROVENCE ALPES COTE D AZUR
94	2A004	0	CORSE

liste_region.csv

3.2 Jointure de tables

Lisez le paragraphe **Jointures** du milieu de la p. 156 à la p. 157

- Comme support à cette question, télechargez et imprimez le fichier support_question_59.pdf (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/support_question_59.pdf)

Télécharger la table des départements (fichier *departement2019.csv*) suivant et l'enregistrer dans le dossier contenant ce fichier jupyter.
En-dessous de la ligne des champs, cette table contient 101 enregistrements (96 départements de métropole et 5 départements d'outre-mer).
Table des départements (http://www.famille-reboul.fr/NSI/seq_11/departement2019.csv)

dep	reg	cheflieu	tncc	ncc
1	84	1053	5	AIN
2	32	2408	5	AISNE
3	84	3190	5	ALLIER
4	93	4070	4	ALPES DE HAUTE PROVENCE
5	93	5061	4	HAUTES ALPES
6	93	6088	4	ALPES MARITIMES
7	84	7186	5	ARDECHE
8	44	8105	4	ARDENNES
9	76	9122	5	ARIEGE
10	44	10387	5	AUBE
11	76	11069	5	AUDE
12	76	12202	5	AVEYRON
13	93	13055	4	BOUCHES DU RHONE
14	28	14118	2	CALVADOS
15	84	15014	2	CANTAL
16	75	16015	3	CHARENTE
17	75	17300	3	CHARENTE MARITIME
18	24	18033	2	CHER
19	75	19272	3	CORREZE
21	27	21231	3	COTE D OR
22	53	22278	4	COTES D ARMOR
23	75	23096	3	CREUSE
24	75	24322	3	DORDOGNE
25	27	25056	2	DOUBS
26	84	26362	3	DROME
27	28	27229	5	EURE
28	24	28085	1	EURE ET LOIR
29	53	29232	2	FINISTERE

2A	94	2A004	3	CORSE DU SUD
2B	94	2B033	3	HAUTE CORSE
30	76	30189	2	GARD
31	76	31555	3	HAUTE GARONNE
32	76	32013	2	GERS
33	75	33063	3	GIRONDE

departement2019.csv

Pour chaque ligne du fichier *département2019.csv* est indiqué un numéro de région (champ 'reg').
Le but est de créer une table où, pour chaque département, apparait un champ supplémentaire contenant le nom de la région.
C'est donc une jointure qui est demandée.

59) Créer une fonction qui effectue cette jointure. Seront passés en paramètres :

- la table *departement2019* correspondant au fichier departement2019.csv
- L'index du champ 'reg' dans la table *departement2019*
- La table *liste_region* créée à la question 58 sous la forme liste de listes. Sa première ligne est la liste des champs.
- L'index du champ 'reg' dans la table *liste_region*
- L'index du champ 'ncc' dans la table *liste_region*
  (c'est le nom de la region).

  Sera affichée à la fin la table *liste_jointure* résultat de la jointure.

Résultat attendu :

```
liste_jointure =  [['1', '84', '1053', '5', 'AIN', 'AUVERGNE RHONE ALPES'],
['2', '32', '2408', '5', 'AISNE', 'HAUTS DE FRANCE'], ['3', '84', '3190',
'5', 'ALLIER', 'AUVERGNE RHONE ALPES'], ['4', '93', '4070', '4', 'ALPES DE
HAUTE PROVENCE', 'PROVENCE ALPES COTE D AZUR'], ['5', '93', '5061', '4', 'H
AUTES ALPES', 'PROVENCE ALPES COTE D AZUR'], ['6', '93', '6088', '4', 'ALPE
S MARITIMES', 'PROVENCE ALPES COTE D AZUR'], ['7', '84', '7186', '5', 'ARDE
CHE', 'AUVERGNE RHONE ALPES'], ['8', '44', '8105', '4', 'ARDENNES', 'GRAND
EST'], ['9', '76', '9122', '5', 'ARIEGE', 'OCCITANIE'], ['10', '44', '1038
7', '5', 'AUBE', 'GRAND EST'], ['11', '76', '11069', '5', 'AUDE', 'OCCITANI
E'], ['12', '76', '12202', '5', 'AVEYRON', 'OCCITANIE'], ['13', '93', '1305
5', '4', 'BOUCHES DU RHONE', 'PROVENCE ALPES COTE D AZUR'], ['14', '28', '1
```

```
4118 ,  2 ,  CALVADOS ,  NORMANDIE ], [ 15 ,  84 ,  15014 ,  2 ,  CANTAL ,
'AUVERGNE RHONE ALPES'], ['16', '75', '16015', '3', 'CHARENTE', 'NOUVELLE A
QUITAINE'], ['17', '75', '17300', '3', 'CHARENTE MARITIME', 'NOUVELLE AQUIT
AINE'], ['18', '24', '18033', '2', 'CHER', 'CENTRE VAL DE LOIRE'], ['19',
'75', '19272', '3', 'CORREZE', 'NOUVELLE AQUITAINE'], ['21', '27', '21231',
'3', 'COTE D OR', 'BOURGOGNE FRANCHE COMTE'], ['22', '53', '22278', '4', 'C
OTES D ARMOR', 'BRETAGNE'], ['23', '75', '23096', '3', 'CREUSE', 'NOUVELLE
AQUITAINE'], ['24', '75', '24322', '3', 'DORDOGNE', 'NOUVELLE AQUITAINE']
```

Liste de listes "liste_jointure" obtenue à la fin de votre programme

In [ ]:

```python
from copy import deepcopy


def jointure(table1, index_ch1, table2, index_ch2, index_champs_ajout
):
    """
    table1 est la première table sans la liste des champs en première
ligne.
    Dans cette question, table1 est la liste_departements
    index_ch1 est l'indice du champ à tester dans table1.

    table2 est la deuxième table sans la liste des champs en première
ligne.
    Dans cette question, table2 est la liste_region_sans_champs
    index_ch2 est l'indice du champ à tester dans table2.

    index_champs_ajout est le champ qu'il faut ajouter à table1 (pour
la jointure)

    Parametres nommes
    -----------------
    table1 : Liste de listes de str
    index_ch1 : int
    table2 : Liste de listes de str
    index_ch2 : int
    index_champs_ajout : int

    Retourne
    --------
    rep : Table qui est la jointure de table 1 et du champ de table2
dont l'index est index_champs_ajout

    """

    rep = deepcopy(table1)
```

```
    ...
    ...
    ...
    ...
    return dep



#  -- 1. CREATION DE liste_departements (pour TABLE1) par conversion du fichier département2019.csv -- #
#  Téléchargement du fichier
f_departements = open("departement2019.csv","r",encoding="utf-8")

#  Conversion en tables des valeurs
champs_dep = f_departements.readline()  #  Lecture de la première ligne
lignes_dep = f_departements.readlines() #  Lecture des autres lignes

#  Création de la liste de listes des départements.
liste_departements  = []
for ligne in lignes_dep:
    liste = ligne.rstrip().split(',')
    liste_departements.append(liste)

#  Création de la liste des champs des départements
listes_champs_dep = champs_dep.rstrip().split(',')



#  -- 2. CREATION DE l'indice du champ commun 'reg' dans la table des régions (pour index_ch1) -- #
index_reg_dep = listes_champs_dep.index('reg')



#  -- 3. CREATION DE listes_region_copy (pour TABLE2) par copie de la liste originale liste_region -- #
liste_region_copy = deepcopy(liste_region)   #  Permet de garder intacte la liste originale.
listes_champs_reg = liste_region_copy.pop(0)  #  Détache la 1ere ligne qui est la liste des champs.



#  -- 4. CREATION de l'indice du champ commun 'reg' dans la table des régions (pour index_ch2) -- #
index_reg_region = listes_champs_reg.index('reg')
```

```
# -- 5. CREATION de l'indice du champ 'ncc' qui contient le nom des r
égions (pour index_champs_ajout) -- #
index_reg_nom = listes_champs_reg.index('ncc')



#  -- APPEL de la fonction jointure avec le 5 paramètres précédemment
créés -- #
liste_jointure = jointure(liste_departements, index_reg_dep, liste_re
gion_copy, index_reg_region, index_reg_nom )
print("liste_jointure = ", liste_jointure)



#  Script qui utilise la bibliothèque csv pour transformer une liste
de listes en un fichier csv.

import csv

#  Ouverture du fichier avec 'with' ce qui évite de gérer la fermetur
e.
with open('liste_jointure.csv', 'w', newline='') as f:  #  newline=''
évite des sauts de lignes en plus.
    writer = csv.writer(f, delimiter=',', quotechar='|', quoting=csv.
QUOTE_MINIMAL)  #  Fabrique le writer.
    writer.writerows(liste_jointure)  #  writerows écrit en une seule
fois toute la liste de listes.
```

liste_jointure convertie en fichier csv et ouvert dans un tableur :

1	84	1053	5	AIN	AUVERGNE RHONE ALPES
2	32	2408	5	AISNE	HAUTS DE FRANCE
3	84	3190	5	ALLIER	AUVERGNE RHONE ALPES
4	93	4070	4	ALPES DE HAUTE PROVENCE	PROVENCE ALPES COTE D AZUR
5	93	5061	4	HAUTES ALPES	PROVENCE ALPES COTE D AZUR
6	93	6088	4	ALPES MARITIMES	PROVENCE ALPES COTE D AZUR
7	84	7186	5	ARDECHE	AUVERGNE RHONE ALPES
8	44	8105	4	ARDENNES	GRAND EST
9	76	9122	5	ARIEGE	OCCITANIE
10	44	10387	5	AUBE	GRAND EST
11	76	11069	5	AUDE	OCCITANIE
12	76	12202	5	AVEYRON	OCCITANIE
13	93	13055	4	BOUCHES DU RHONE	PROVENCE ALPES COTE D AZUR
14	28	14118	2	CALVADOS	NORMANDIE
15	84	15014	2	CANTAL	AUVERGNE RHONE ALPES

16	75	16015	3	CHARENTE	NOUVELLE AQUITAINE
17	75	17300	3	CHARENTE MARITIME	NOUVELLE AQUITAINE
18	24	18033	2	CHER	CENTRE VAL DE LOIRE
19	75	19272	3	CORREZE	NOUVELLE AQUITAINE
21	27	21231	3	COTE D OR	BOURGOGNE FRANCHE COMTE
22	53	22278	4	COTES D ARMOR	BRETAGNE
23	75	23096	3	CREUSE	NOUVELLE AQUITAINE
24	75	24322	3	DORDOGNE	NOUVELLE AQUITAINE
25	27	25056	2	DOUBS	BOURGOGNE FRANCHE COMTE
26	84	26362	3	DROME	AUVERGNE RHONE ALPES
27	28	27229	5	EURE	NORMANDIE
28	24	28085	1	EURE ET LOIR	CENTRE VAL DE LOIRE
29	53	29232	2	FINISTERE	BRETAGNE
2A	94	2A004	3	CORSE DU SUD	CORSE
2B	94	2B033	3	HAUTE CORSE	CORSE
30	76	30189	2	GARD	OCCITANIE
31	76	31555	3	HAUTE GARONNE	OCCITANIE
32	76	32013	2	GERS	OCCITANIE

liste_jointure.csv

