

Création d'une table à partir d'un fichier CSV

On dispose d'un fichier `occitanie.csv`.

Ouvert avec le Bloc-notes on obtient ceci :

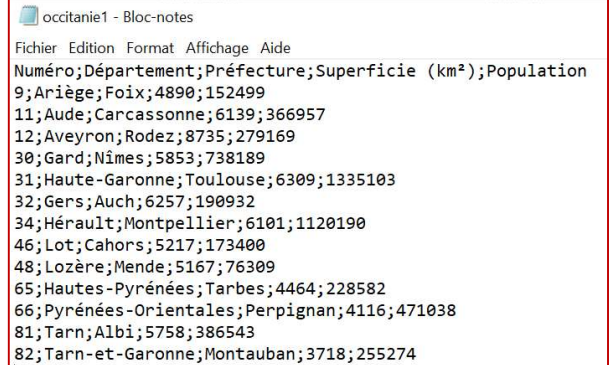
Le format csv

Le sigle **CSV** signifie **Comma-Separated Values**


Les fichiers CSV sont des fichiers **au format texte**

Ils représentent **des données tabulaires**.


Les valeurs sont séparées par un **séparateur**, en général une **virgule** ou un **point-virgule** (ou deux points ...)



```
occitanie1 - Bloc-notes
Fichier Edition Format Affichage Aide
Numéro;Département;Préfecture;Superficie (km²);Population
9;Ariège;Foix;4890;152499
11;Aude;Carcassonne;6139;366957
12;Aveyron;Rodez;8735;279169
30;Gard;Nîmes;5853;738189
31;Haute-Garonne;Toulouse;6309;1335103
32;Gers;Auch;6257;190932
34;Hérault;Montpellier;6101;1120190
46;Lot;Cahors;5217;173400
48;Lozère;Mende;5167;76309
65;Hautes-Pyrénées;Tarbes;4464;228582
66;Pyrénées-Orientales;Perpignan;4116;471038
81;Tarn;Albi;5758;386543
82;Tarn-et-Garonne;Montauban;3718;255274
```

 Avec Python on saisit le code suivant :

```
fich=open("occitanie1.csv","r") #on ouvre en lecture le fichier occitanie.csv
champs=fich.readline() # champs prend pour valeur la première ligne
lignes=fich.readlines() # lignes prend pour valeur le reste des lignes
fich.close() # On ferme le fichier
print(champs)
print(lignes)
```

 Les affichages sont les suivants :

```
Numéro;Département;Préfecture;Superficie (km²);Population

['9;Ariège;Foix;4890;152499\n', '11;Aude;Carcassonne;6139;366957\n', '12;Aveyron;Rodez;8735;279169\n', '30;Gard;Nîmes;5853;738189\n', '31;Haute-Garonne;Toulouse;6309;1335103\n', '32;Gers;Auch;6257;190932\n', '34;Hérault;Montpellier;6101;1120190\n', '46;Lot;Cahors;5217;173400\n', '48;Lozère;Mende;5167;76309\n', '65;Hautes-Pyrénées;Tarbes;4464;228582\n', '66;Pyrénées-Orientales;Perpignan;4116;471038\n', '81;Tarn;Albi;5758;386543\n', '82;Tarn-et-Garonne;Montauban;3718;255274\n']
```

1) La variable `champs` est :

- ☐ une chaînes de caractères
- ☐ une liste de chaînes de caractères
- ☐ un tuple
- ☐ un dictionnaire
- ☐ autre

2) La variable `lignes` est :

- ☐ une chaînes de caractères
- ☐ une liste de chaînes de caractères
- ☐ un tuple
- ☐ un dictionnaire
- ☐ autre

On continue le code par les lignes suivantes :

3) Que permet de faire la ligne `print(5*"\\n")` ?

Cela permet d'afficher 5 sauts de lignes

```
print(5*"\\n")
fich=open("occitanie1.csv","r")
champs=fich.readline()
table=[ligne for ligne in fich]
fich.close()
print(table)
```

4) La ligne de code `table=[ligne for ligne in fich]` définit la variable `table` comme étant :

- ☐ une chaînes de caractères
- ☐ une liste
- ☐ un tuple
- ☐ un dictionnaire
- ☐ un booléen
- ☐ un entier

5) Cette façon de « construire » la variable `table` s'appelle :


- ☐ par insertion
- ☐ par fusion
- ☐ par compréhension
- ☐ par lecture
- ☐ par instruction

 On obtient l'affichage suivant

```
['9;Ariège;Foix;4890;152499\n', '11;Aude;Carcassonne;6139;366957\n', '12;Aveyron;Rodez;8735;279169\n', '30;Gard;Nîmes;5853;738189\n', '31;Haute-Garonne;Toulouse;6309;1335103\n', '32;Gers;Auch;6257;190932\n', '34;Hérault;Montpellier;6101;1120190\n', '46;Lot;Cahors;5217;173400\n', '48;Lozère;Mende;5167;76309\n', '65;Hautes-Pyrénées;Tarbes;4464;228582\n', '66;Pyrénées-Orientales;Perpignan;4116;471038\n', '81;Tarn;Albi;5758;386543\n', '82;Tarn-et-Garonne;Montauban;3718;255274\n']
```


6) Que constate-t-on ? :

On obtient exactement la même chose.

 On modifie la ligne de code :

```
table=[ligne for ligne in fich] par:  
table=[ligne.split(";") for ligne in fich]
```

```
fich=open("occitanie1.csv","r")  
champs=fich.readline()  
table=[ligne.split(";") for ligne in fich]  
fich.close()  
print(table)
```


 On obtient alors l'affichage suivant :

```
[[['9', 'Ariège', 'Foix', '4890', '152499\n'], ['11', 'Aude', 'Carcassonne', '6139', '366957\n'], ['12', 'Aveyron', 'Rodez', '8735', '279169\n'], ['30', 'Gard', 'Nîmes', '5853', '738189\n'], ['31', 'Haute-Garonne', 'Toulouse', '6309', '1335103\n'], ['32', 'Gers', 'Auch', '6257', '190932\n'], ['34', 'Hérault', 'Montpellier', '6101', '1120190\n'], ['46', 'Lot', 'Cahors', '5217', '173400\n'], ['48', 'Lozère', 'Mende', '5167', '76309\n'], ['65', 'Hautes-Pyrénées', 'Tarbes', '4464', '228582\n'], ['66', 'Pyrénées-Orientales', 'Perpignan', '4116', '471038\n'], ['81', 'Tarn', 'Albi', '5758', '386543\n'], ['82', 'Tarn-et-Garonne', 'Montauban', '3718', '255274\n']]]
```


7) Quelle est la nature de la variable table ?

La méthode **.split(argument)**

La méthode `split(car)` s'applique à une chaîne de caractères qu'elle transforme en liste de chaînes. Le caractère `car` permet de déterminer là où il faut « couper » la chaîne pour déterminer les différents items de la liste. Par défaut (si `car` n'est pas précisé) c'est un espace.

 On modifie la ligne de code :

```
table=[ligne.split(";") for ligne in fich] par:  
table=[ligne.rstrip().split(";") for ligne in fich]
```

 On obtient alors l'affichage suivant :

```
[[['9', 'Ariège', 'Foix', '4890', '152499'], ['11', 'Aude', 'Carcassonne', '6139', '366957'], ['12', 'Aveyron', 'Rodez', '8735', '279169'], ['30', 'Gard', 'Nîmes', '5853', '738189'], ['31', 'Haute-Garonne', 'Toulouse', '6309', '1335103'], ['32', 'Gers', 'Auch', '6257', '190932'], ['34', 'Hérault', 'Montpellier', '6101', '1120190'], ['46', 'Lot', 'Cahors', '5217', '173400'], ['48', 'Lozère', 'Mende', '5167', '76309'], ['65', 'Hautes-Pyrénées', 'Tarbes', '4464', '228582'], ['66', 'Pyrénées-Orientales', 'Perpignan', '4116', '471038'], ['81', 'Tarn', 'Albi', '5758', '386543'], ['82', 'Tarn-et-Garonne', 'Montauban', '3718', '255274']]]
```

8) Quelle est la nature de la variable table ? qu'est-ce qui a changé ?

La méthode **.rstrip(argument)**

La méthode `rstrip(car)` s'applique à une chaîne de caractères. Elle supprime les caractères `car` situés en fin de ligne. Par défaut c'est un espace. La méthode `lstrip(car)` fait la même chose pour les caractères situés en début.

Applications

Compléter chaque copie d'écran :

Ap1

```
>>> txt="le chat dort"  
>>> txt.split()  
['le', 'chat', 'dort']
```

Ap2

```
>>> L=["nom;prenom;classe","Azer;Joe;Ce2"]  
>>> L[1].split(";")  
['Azer', 'Joe', 'Ce2']
```

Ap3


```
>>> txt="Yeah!!!!!"  
>>> txt.rstrip("!")  
'Yeah'  
>>> txt  
'Yeah!!!!!!'
```

Ap4


```
>>> txt="## Ceci est un commentaire."  
>>> txt.split()  
['##', 'Ceci', 'est', 'un', 'commentaire.']  
>>> txt.lstrip("#").rstrip(".").split()  
['Ceci', 'est', 'un', 'commentaire']
```

A partir de maintenant on dispose d'une liste de listes.

Problème : Les fichiers CSV étant au format texte, toutes les données sont des chaînes de caractères. Or ici la superficie et la population sont des nombres (ici des entiers).

 Il nous faut donc convertir ces données en entiers.
Il suffit alors de lire table, listes par listes (ligne par ligne) et pour chaque liste, convertir les éléments de rang 3 et 4 en entier. On saisit le code :

```
for elt in table:
    elt[3]=int(elt[3])
    elt[4]=int(elt[4])
print(table)
```

9)  Quelle est alors le début de l'affichage ?


```
[['9', 'Ariège', 'Foix', 4890, 152499], ['11', 'Aude',
```

Synthèse

On aurait pu tout faire d'un coup avec le code suivant :

```
fich=open("occitanie1.csv","r") #on ouvre en lecture le fichier occitanie.csv
champs=fich.readline() # champs prend pour valeur la première ligne
lignes=fich.readlines() # lignes est une liste de chaînes de caractères
fich.close() # On ferme le fichier
# On crée la table à l'aide d'une boucle et non par compréhension
table=[] # on initialise une table vide
for ligne in lignes : # On crée la table à l'aide d'une boucle
    liste=ligne.rstrip().split(";") #chaque item de lignes est converti en liste
    liste[3]=int(liste[3]) # on convertit au bon type
    liste[4]=int(liste[4])
    table.append(liste) # on ajoute cette liste à la table
```

Recherche d'éléments dans une table par critère

 On veut déterminer les départements dont la population est supérieure à 500 000 habitants.
On va alors saisir le code :

```
rep=[]
for ligne in table:
    if ligne[4]>500000:
        rep.append(ligne)
```

 On obtient l'affichage

```
[['30', 'Gard', 'Nîmes', 5853, 738189], ['31', 'Haute-Garonne', 'Toulouse', 6309, 1335103], ['34', 'Hérault', 'Montpellier', 6101, 1120190]]
```

On veut maintenant le même critère mais ne conserver que le nom du département et sa population.

 On veut donc obtenir pour la variable rep :

```
[['Gard', 738189], ['Haute-Garonne', 1335103], ['Hérault', 1120190]]
```

10)  Compléter alors le code pour obtenir ce résultat :

```
rep=[]
for ligne in table:
    if ligne[4]>500000:
        L=[ligne[1],ligne[4]]
        rep.append(L)
print(rep)
```

Travail à effectuer

- 1) Ouvrir le fichier **france.csv** avec le Bloc-notes et noter les différents champs ainsi que leur type (chaîne, entier, booléen, flottant ...)

Les différents champs de **france.csv** sont :

Nom du champ

Type

- 2) Ecrire le code permettant de convertir ce fichier en une table, nommée **table** dont chaque élément a le bon type.
- 3) Créer alors par recherche la liste des départements dont la population est supérieure à 500 000 habitants. On demande juste de retourner une liste dont les items sont les noms des départements. On nommera cette liste **dept_plus**.
- 4) Créer alors par recherche la liste des départements dont la superficie est supérieure à 6000 km² et la population est supérieure à 500 000 habitants. On ne gardera que le nom du département, sa région, sa population et sa superficie. On renverra donc une table (liste de listes). On nommera cette table **ma_selection**.
- 5) Créer alors par recherche la liste des départements de l'Occitanie. On ne gardera que le numéro, nom du département et la préfecture. On renverra donc une table (liste de listes). On nommera cette table **occitanie**.