Traitement des données en tables

qkzk

2021/05/25

En bref : *Le format CSV est couramment utilisé pour échanger des données habituellement traitées à l'aide de tableurs ou de logiciels de bases de données principalement. Nous allons apprendre à importer et exporter des données en utilisant ce format.*

1. Données en table

De quoi parle-t-on ?

Par "données en table" on entend ce type d'information :

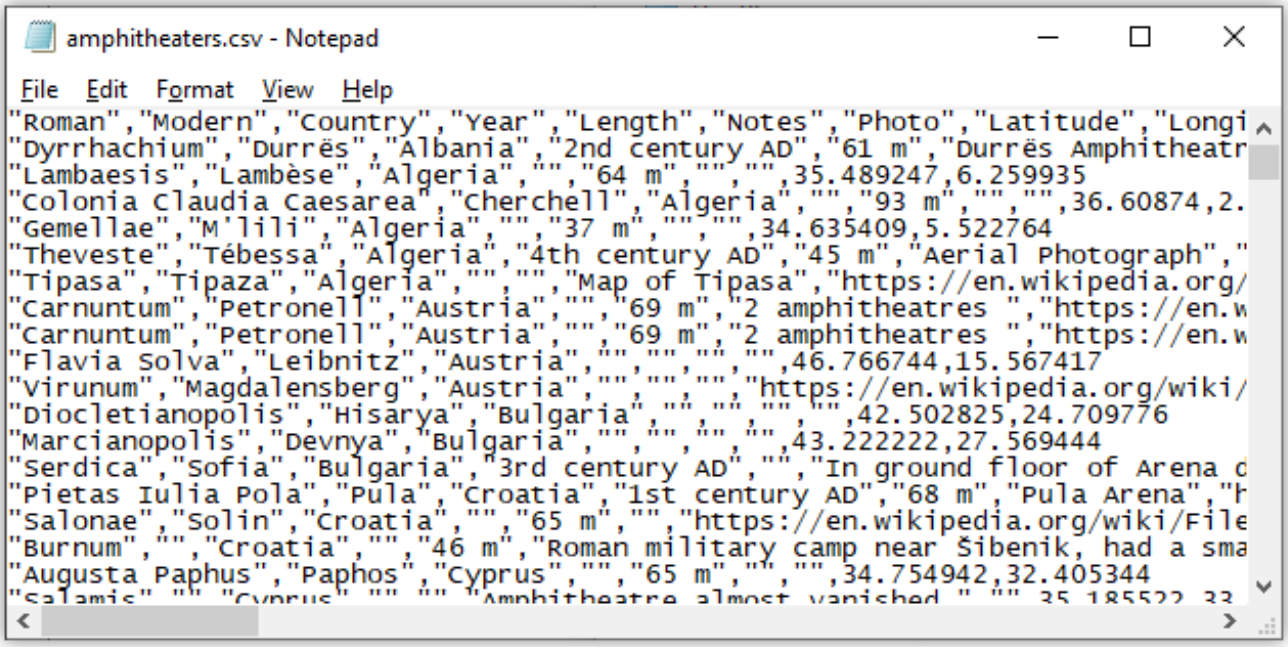
- correctement importé dans un tableur

	A	B	C	D	E	F	G	H	I
1	Code client	Pièce n°	Type	Date d'émission	Montant TTC(EUR)	Montant de Transaction TTC	Devise de Transaction	Date d'échéance	Solde TTC(EUR)
2	730528	J582749	FAC	06/02/2015	1172,2	1 407	EUR	15/04/2015	1 407
3	730528	P521972	FAC	17/02/2015	855	1 026	EUR	15/04/2015	1 026
4	730528	P521973	FAC	17/02/2015	261,8	314	EUR	15/04/2015	314
5	730528	P568793	FAC	09/03/2015	1204	1 445	EUR	15/05/2015	1 445
6	758156	P545455	FAC	02/03/2015	1173,63	1 408	EUR	15/05/2015	1 408
7	758156	P545457	FAC	02/03/2015	270,6	325	EUR	15/05/2015	325
8	758156	P553989	FAC	04/03/2015	482,2	579	EUR	15/05/2015	579
9	758156	P561049	FAC	06/03/2015	1150,58	1 381	EUR	15/05/2015	1 381
10	758156	P573339	TRT	10/03/2015	286,56	344	EUR	15/05/2015	344
11	700078	P555633	FAC	05/03/2015	849,82	1 020	EUR	15/05/2015	1 020
12	600334	P490079	FAC	10/02/2015	1691,46	2 030	EUR	15/04/2015	2 030
13	705930	P555849	FAC	05/03/2015	845,6	1 015	EUR	15/05/2015	1 015
14	743022	P536894	FAC	25/02/2015	374,14	449	EUR	15/04/2015	449
15	743022	P542359	FAC	27/02/2015	331,6	398	EUR	15/04/2015	398
16	756596	P553987	FAC	04/03/2015	610,6	733	EUR	15/05/2015	733
17	706588	P519692	FAC	16/02/2015	2995,2	3 594	EUR	15/04/2015	3 594
18	706588	P519693	FAC	16/02/2015	194,23	233	EUR	15/04/2015	233
19	706588	P572795	FAC	10/03/2015	174,1	209	EUR	15/05/2015	209
20	706588	P572796	FAC	10/03/2015	194,5	233	EUR	15/05/2015	233
21	719382	P553297	FAC	04/03/2015	1249,85	1 500	EUR	10/04/2015	1 500

- sans mise en forme dans un tableur

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	code postal,"insee","article","ville","ARTICLE","VILLE","libelle","region","nom region","dep","nom dep","longitude","latitude","codex","metaphone"													
2	01500,"01004","","AmbÃ©rieu-en-Bugey","","AMBERIEU-EN-BUGEY","AMBERIEU EN BUGEY","82","RHONE-ALPES","01","Ain","45.979851","5.33689","A516","AMPRNPJ"													
3	01330,"01005","","AmbÃ©rieux-en-Dombes","","AMBERIEUX-EN-DOMBES","AMBERIEUX EN DOMBES","82","RHONE-ALPES","01","Ain","45.998057","4.902498","A516","AMPRKSNTMP													
4	01300,"01006","","AmblÃ©on","","AMBLEON","AMBLEON","82","RHONE-ALPES","01","Ain","45.74987","5.601326","A514","AMPLN"													
5	01500,"01007","","Ambronay","","AMBRONAY","AMBRONAY","82","RHONE-ALPES","01","Ain","46.00648","5.35999","A516","AMPRN"													
6	01500,"01008","","Ambutrix","","AMBUTRIX","AMBUTRIX","82","RHONE-ALPES","01","Ain","45.939594","5.338195","A513","AMPTRKS"													
7	01300,"01009","","Andert-et-Condon","","ANDERT-ET-CONDON","ANDERT ET CONDON","82","RHONE-ALPES","01","Ain","45.794688","5.655624","A536","ANTRTTKNTN"													
8	01350,"01010","","Anglefort","","ANGLEFORT","ANGLEFORT","82","RHONE-ALPES","01","Ain","45.91333","5.808992","A524","ANKLFRT"													
9	01100,"01011","","Apremont","","APREMONT","APREMONT","82","RHONE-ALPES","01","Ain","46.207172","5.657787","A165","APRMNT"													
10	01110,"01012","","Aranc","","ARANC","ARANC","82","RHONE-ALPES","01","Ain","46.003679","5.508627","A652","ARNK"													
11	01230,"01013","","Arandas","","ARANDAS","ARANDAS","82","RHONE-ALPES","01","Ain","45.89596","5.488735","A653","ARNTS"													
12	01100,"01014","","Arbent","","ARBENT","ARBENT","82","RHONE-ALPES","01","Ain","46.295735","5.6821","A615","ARPNT"													
13	01300,"01015","","Arbignieu","","ARBIGNIEU","ARBIGNIEU","82","RHONE-ALPES","01","Ain","45.728482","5.650354","A612","ARPN"													
14	01190,"01016","","Arbigny","","ARBIGNY","ARBIGNY","82","RHONE-ALPES","01","Ain","46.469456","4.961984","A612","ARPN"													
15	01230,"01017","","Argis","","ARGIS","ARGIS","82","RHONE-ALPES","01","Ain","45.933505","5.491159","A622","ARJS"													
16	01510,"01019","","Armix","","ARMIX","ARMIX","82","RHONE-ALPES","01","Ain","45.849253","5.585465","A652","ARMKS"													
17	01480,"01021","","Ars-sur-Formans","","ARS-SUR-FORMANS","ARS SUR FORMANS","82","RHONE-ALPES","01","Ain","45.993535","4.82113","A626","ARSSRFRMNS"													
18	01510,"01022","","Artemare","","ARTEMARE","ARTEMARE","82","RHONE-ALPES","01","Ain","45.871579","5.691073","A635","ARTMR"													
19	01570,"01023","","AsniÃ¨res-sur-SaÃ´ne","","ASNIERES-SUR-SAONE","ASNIERES SUR SAONE","82","RHONE-ALPES","01","Ain","46.383037","4.883211","A256","ASNRSSRSN"													
20	01340,"01024","","Attignat","","ATTIGNAT","ATTIGNAT","82","RHONE-ALPES","01","Ain","46.28472","5.16116","A325","ATNT"													
21	01380,"01025","","BÃ©gÃ©-la-Ville","","BAGE-LA-VILLE","BAGE LA VILLE","82","RHONE-ALPES","01","Ain","46.316575","4.946254","B241","PJLFL"													
22	01380,"01026","","BÃ©gÃ©-le-ChÃ¢tel","","BAGE-LE-CHATEL","BAGE LE CHATEL","82","RHONE-ALPES","01","Ain","46.30863","4.930055","B242","PJLXTL"													
23	01360,"01027","","Balan","","BALAN","BALAN","82","RHONE-ALPES","01","Ain","45.833836","5.097323","B450","PLN"													
24	01990,"01028","","Baneins","","BANEINS","BANEINS","82","RHONE-ALPES","01","Ain","46.110265","4.90141","B552","PNNS"													
25	01270,"01029","","Beaupont","","BEAUPONT","BEAUPONT","82","RHONE-ALPES","01","Ain","46.408616","5.257749","B153","PPNT"													

- sans mise en forme dans un fichier texte



Excel et les tableurs évolués réalisent un traitement complexe des données mais on peut toujours exporter un tableau de données vers un fichier au format particulier ou importer un tel fichier.

2. Enregistrements

- Un **enregistrement** est une structure de données, de types éventuellement différents, auxquelles ont accède grâce à un nom.

  Exemple : On peut représenter les notes d'un élève dans différentes disciplines à l'aide d'un enregistrement :

  ```
  {'Nom' : 'Joe', 'Anglais': '17', 'Info': '18', 'Maths': '16'}
  ```

  Les champs (ou clés ou attributs) de ces enregistrements sont ici `'Nom'`, `'Anglais'`, `'Info'` et `'Maths'`. On leur associe des valeurs, ici `'Joe'`, `'17'`, `'18'` et `'16'`.

- En Python, on utilisera principalement les **dictionnaires** pour représenter les enregistrements.

Les données en table dans un langage de programmation

En python, ce tableau peut prendre différentes formes :

1. Tableau doublement indexé :

```
contacts = [
    ['Duchmol', 'Robert', 'robert@email.com',],
    ['Lemeilleur', 'Franky', 'franky@email.com',],
    ['Poivre', 'Jacques', 'jacque@email.com']
]
```

2. Tableau de dictionnaires :

```
contacts = [
    {'Nom': 'Duchmol', 'Prénom', 'Robert', 'email': 'robert@email.com'},
    {'Nom': 'Lemeilleur', 'Prénom', 'Franky', 'email': 'franky@email.com'},
    {'Nom': 'Poivre', 'Prénom', 'Jacques', 'email': 'jacque@email.com'}
]
```

Le contenu ne change pas, les manipulations nécessaires pour y accéder si.

Dans certains cas on dispose directement des données (via l'énoncé ou un travail préalable). Parfois on doit importer les données depuis un fichier csv.

Comparaison des formats

Dans le premier cas on accède aux données comme dans un tableau 2D :

- accéder au premier enregistrement.

```
>>> contacts[0]
['Nom': 'Duchmol', 'Prénom', 'Robert', 'email': 'robert@email.com']
```

- quel est le prénom du second contact ?

```
>>> contacts[1][1]
'Franky'
```

Dans le second cas, on accède toujours aux enregistrements par leur indice mais on accède aux valeurs *par leur clés.*

- quel est le prénom du second contact ?

```
>>> contacts[1]['Prénom']
'Franky'
```

3. Fichiers CSV

- Le format CSV (Comma Separated Value) est couramment utilisé pour importer ou exporter des données d'une feuille de calcul d'un tableur. C'est un fichier texte dans lequel chaque ligne correspond à une ligne du tableau, les colonnes étant séparées par des **virgules**. Il permet donc de représenter une liste d'enregistrements ayant les mêmes champs.

  À noter : *Ce format est né bien avant les ordinateurs personnels et le format* `xls` *puisque c'est en 1972 qu'il a été introduit.*

- Pour éviter les problèmes dûs à l'absence de standardisation du séparateur décimal (virgule en France, point dans les pays anglo-saxons), on peut paramétrer son tableur pour utiliser le point pour les nombres (français suisse par exemple).
- Voici un exemple de feuille de calcul.

	A	B	C	D
1	Nom	Anglais	Info	Maths
2	Joe	17	18	16
3	Zoé	15	17	19
4	Max	19	13	14

Le fichier CSV correspondant est :

```
'Nom','Anglais','Info','Maths'
'Joe','17','18','16'
'Zoé','15','17','19'
'Max','19','13','14'
```

4. Lecture de fichiers CSV

- On peut choisir de représenter en Python les fichiers CSV par des listes de dictionnaires dont les clés sont les noms des colonnes.
- Avec l'exemple précédent, on définit la liste :

```
Table1 = [
    {'Nom': 'Joe','Anglais': '17', 'Info': '18', 'Maths': '16'},
    {'Nom': 'Zoé','Anglais': '15', 'Info': '17', 'Maths': '19'},
    {'Nom': 'Max','Anglais': '19', 'Info': '13', 'Maths': '14'}]
```

À noter : *En utilisant le vocabulaire habituel décrivant une feuille de calcul d'un tableur :*

- *une table est une liste de dictionnaires: ici* `Table1`
- *une ligne est un dictionnaire, ici* `Table1[0]`
- *une cellule est une valeur associée à une clé d'un dictionnaire, ici* `Table1[0]['Info']`

5. Import d'un fichier CSV

La méthode à retenir

- pour l'import on crée une liste de dictionnaires (un par ligne de la table). La première ligne du fichier CSV est considérée comme la ligne des noms des attributs. `fichier` est une chaîne de caractères donnant le nom du fichier. Par exemple : `depuis_csv('Ma_base.csv')` pour charger le fichier `Ma_base.csv`

```
import csv
def depuis_csv(fichier):
    with open(fichier, 'r') as contenu_csv:
        lecteur = csv.DictReader(contenu_csv)
        return [dict(ligne) for ligne in lecteur]
```

6. Exporter vers un fichier CSV avec Python

- Pour l'export, on entre le nom de la table sous forme de chaîne. On donne l'ordre des colonnes sous forme d'une liste d'attributs.

```
def vers_csv(nom_fichier, ordre):
    with open(nom_fichier, 'w') as contenu:
        writer = csv.DictWriter(contenu, fieldnames=ordre)
        table = eval(nom)
        writer.writeheader()  # pour la 1ere lire, celle des attributs
        for ligne in table:
            writer.writerow(ligne)  # ajoute les lignes de la table
```

- Pour exporter la table `Table1` on utilise :

```
>>> vers_csv('Table1.csv', ['Nom', 'Anglais', 'Info', 'Maths'])
```

Python crée alors un fichier `Table1.csv` au format CSV :

```
Nom,Anglais,Info,Maths
Joe,17,18,16
Zoé,15,17,19
Max,19,13,14
```

6. Sélectionner et projeter

On est régulièrement amené à récupérer une partie des données.

- lorsqu'on souhaite accéder à un ou plusieurs enregistrements vérifiant un critère, on réalise une *sélection*.
- lorsqu'on souhaite accéder à toutes les données d'une colonne on réalise une *projection*.

Exemple de sélection.

Supposons qu'on dispose d'une table enregistrée dans une liste de dictionnaires :

```
Table1 = [
    {'Nom': 'Joe','Anglais': '17', 'Info': '18', 'Maths': '16'},
    {'Nom': 'Zoé','Anglais': '15', 'Info': '17', 'Maths': '19'},
    {'Nom': 'Max','Anglais': '19', 'Info': '13', 'Maths': '14'},
    {'Nom': 'Bob','Anglais': '12', 'Info': '16', 'Maths': '10'}
]
```

On souhaite extraire la liste des enregistrements des élèves ayant eu au moins 16 en maths.

On peut le faire "à la main" :

```
au_moins_16_en_maths = []
for enregistrement in Table1:
    if enregistrement['Maths'] >= 16:
        au_moins_16_en_maths.append(enregistrement)
```

Le résultat est *encore une table* :

```
Table1 = [
    {'Nom': 'Joe','Anglais': '17', 'Info': '18', 'Maths': '16'},
    {'Nom': 'Zoé','Anglais': '15', 'Info': '17', 'Maths': '19'}
]
```

On peut le faire avec une liste par compréhension :

```
au_moins_16_en_maths = [enre for enre in Table1 if enre['Maths'] >= 16]
```

Le résultat est identique.

Exemple de projection

Cette fois, on souhaite récupérer toutes les valeurs pour un champ donné, par exemple toutes les notes de mathématiques.

L'approche est similaire, on crée une liste, on parcourt la table et on ajoute à la liste tous les éléments qui nous intéressent :

- à la main :

```
notes_maths = []
for enregistrement in Table1:
    notes_maths.append(enregistrement['Maths'])
```

- par compréhension :

```
notes_maths = [enre['Maths'] for enre in Table1]
```

Dans les deux cas le résultat est la liste `[16, 19, 14, 10]`

Exercices

1. Adapter la sélection afin de récupérer tous les enregistrements des élèves dont le nom comporte un "o"
2. Projeter afin de construire la liste des noms puis celle des paires de notes d'info et de maths :

```
[(18, 16), (17, 19), (13, 14), (16, 10)]
```

7. *Compléments*

Cette partie est donnée pour votre culture mais n'est pas à apprendre.

Le format CSV n'est pas le seul utilisé pour enregistrer des données. Il existe d'autres formats de fichiers textes mais aussi une approche beaucoup plus élaborée qui remplit le même objectif : les bases de données.

En terminale on étudiera les **bases de données** et en particulier le langage SQL.

Ce sont des programmes qui permettent à plusieurs utilisateurs d'accéder aux ressources, de conserver une trace de ce qui est fait sur les données etc.

Les autres formats texte

Citons :

- CSV, le plus couramment employé, démocratisé par Excel (Microsoft)
- JSON (Javascript Object Notation)

```
{
    "glossary": {
        "title": "example glossary",
        "GlossDiv": {
            "title": "S",
            "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
                    "SortAs": "SGML",
                    "GlossTerm": "Standard Generalized Markup Language",
                    "Acronym": "SGML",
                    "Abbrev": "ISO 8879:1986",
                    "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocB
                        "GlossSeeAlso": ["GML", "XML"]
                    },
                    "GlossSee": "markup"
                }
            }
        }
    }
}
```

Ressemble beaucoup aux `dict` de Python.

- XML (eXtensible Markup Language) - . Syntaxe similaire à l'HTML. Difficile à lire pour un "humain".

```
<!DOCTYPE glossary PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
 <glossary><title>example glossary</title>
  <GlossDiv><title>S</title>
   <GlossList>
    <GlossEntry ID="SGML" SortAs="SGML">
     <GlossTerm>Standard Generalized Markup Language</GlossTerm>
     <Acronym>SGML</Acronym>
     <Abbrev>ISO 8879:1986</Abbrev>
     <GlossDef>
      <para>A meta-markup language, used to create markup
languages such as DocBook.</para>
      <GlossSeeAlso OtherTerm="GML">
      <GlossSeeAlso OtherTerm="XML">
     </GlossDef>
     <GlossSee OtherTerm="markup">
    </GlossEntry>
   </GlossList>
  </GlossDiv>
 </glossary>
```

Mais aussi :

- YAML (yet another markup language)

```yaml
- martin:
    name: Martin D'vloper
    job: Developer
    skills:
      - python
      - perl
      - pascal
- tabitha:
    name: Tabitha Bitumen
    job: Developer
    skills:
      - lisp
      - fortran
      - erlang
```

Contexe

- Généré par un programme : souvent XML ou JSON
- Crée à la main depuis un tableur : CSV
- Utilisé par des machines pour communiquer via une API : JSON
- Pour stocker des configurations de programme : Yaml, INI, format propriétaire

Importer les données en Python sans utiliser le module CSV

Vous rencontrerez vraisemblablement des variantes de la méthode précédente. Nous allons en présenter rapidement plusieurs. Elles ne sont *pas à connaître mais à comprendre.*

Créer un tableau doublement indexé

Voyons d'abord comment réaliser chacune des étapes (c'est un attendu) puis comment le faire à l'aide des outils proposés.

```
CHAMPS = ["nom", "moyenne", "moyenne_brute"]

def lire_fichier_csv(filename, delimiter=','):
  res = []
  with open(filename) as f:
    f.readline() # on zappe la première ligne...
    l = f.readline()
    while l != '':
      args = l.replace('"', '').strip().split(delimiter)
      try:
        enregistrement = []
        for i in range(len(args)):
          enregistrement[i] = args[i]
        res.append(enregistrement)
      except ValueError:
        pass
      l = f.readline()
    return res

liste_csv = lire_fichier_csv('2nde-14-liste.csv', delimiter=';')
```

Commentaires détaillés sur la fonction `lire_fichier_csv`

- ligne 4 : on crée la liste des enregistrements
- ligne 5 : on ouvre le fichier et on le stocke dans une variable temporaire `f`
- ligne 6 : `f.readline()` : lire une ligne et avancer à la suivante
- ligne 7 : on a zappé la première ligne et on lit la seconde

- ligne 5 : tant qu'on a des lignes non vides
- ligne 9 : on nettoie la ligne : virer les ", les espaces de début et de fin et on sépare la ligne tous les 'delimiter'
- ligne 10 : on essaye de créer l'enregistrement
- ligne 11 : c'est une liste python
- ligne 13 : on l'ajoute à la liste des enregistrements
- ligne 16 : si jamais la ligne est imparfaite, on la zappe
- ligne 17 : on avance à la ligne suivante

Qu'obtient-on ? Un tableau doublement indexé. Les descripteurs sont enregistrés dans `CHAMPS`. Ils ne sont pas utilisés dans la fonction d'import.

Variante : vers un tableau de dictionnaire.

Exercice : adapter la première fonction d'import pour qu'elle retourne un tableau de dictionnaires.

Réponse :

```
CHAMPS = ["nom", "moyenne", "moyenne_brute"]

def lire_fichier_csv(filename, delimiter=',', champs=CHAMPS):
  res = []
  with open(filename) as f:
    f.readline() # on zappe la première ligne...
    l = f.readline()
    while l != '':
      args = l.replace('"', '').strip().split(delimiter)
      try:
        enregistrement = {} # on crée un dictionnaire
        for i in range(len(args)):
          enregistrement[champs[i]] = args[i] # les clés sont les éléments de CHAMPS
        res.append(enregistrement)
      except ValueError:
        pass
      l = f.readline()
    return res

liste_csv = lire_fichier_csv('2nde-14-liste.csv', delimiter=';')
```