Chapitre 5 - Données en table

**Compétences** :

* Importer une table depuis un fichier texte tabulé ou un fichier CSV.
* Rechercher les lignes d’une table vérifiant des critères exprimés en logique propositionnelle.
* Trier une table suivant une colonne.
* Construire une nouvelle table en combinant les données de deux tables.

# Introduction

* Une des utilisations principales de l’informatique de nos jours est le **traitement de quantités importantes de données** dans des domaines très variés :
  + *un site de commerce en ligne peut avoir à gérer des bases données pour des dizaines de milliers d’articles en vente, de clients, de commandes ;*  
    *un hôpital doit pouvoir accéder efficacement à tous les détails de traitements de ses patients ;*  
    *etc …*
* Mais si les logiciels de gestion de base de données (SGDB) sont des programmes hautement spécialisés pour effectuer ce genre de tâches le plus efficacement et sûrement possible, il est facile de mettre en œuvre les opérations de base dans un langage de programmation comme Python.

## Données en table

* Organisées en **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**, les données se présentent sous la forme suivante :
* Un **p-uplet nommé** est **un p-uplet**, dont les composantes sont appelées via un **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** au lieu d’un indice. Le principal intérêt de ce type est d’améliorer la lisibilité du code, et partant de réduire les risques d’erreurs.
* Le type des **p-uplets nommés** n’existe pas nativement dans Python. Voici un exemple pour montrer la syntaxe :

MrPotter = {

"nom": "Harry Potter",

"age": 10,

"maison" : "Gryffondor“

}

|  |  |  |
| --- | --- | --- |
| **ID** | **nom** | **prenom** |
| 458 | Naymar | Jean |
| 687 | Javel | Aude |
| 545 | Cover | Harry |
| 102 | Zeblouze | Agathe |

# Le format CSV

* Le format **CSV** (pour***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***, soit en français *valeurs séparées par des virgules*) est un format très pratique pour représenter des données structurées.
* Dans ce format, chaque ligne représente un **enregistrement** et, sur une même ligne, les différents **champs** de l’enregistrement sont séparés par une virgule (d’où le nom).
* Pour éviter les problèmes dus 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

# Notre « fil rouge »

* Dans la suite, nous allons utiliser deux fichiers nommés **countries.csv**  et **cities.csv** qui contiennent quelques données sur les différents pays et villes du monde.
* Les fichiers CSV peuvent être manipulé par un éditeur de texte simple Notepad ou Notepad++ ou par des éditeurs évolués comme Excel

Avec Notepad++ :

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

Avec Excel :

Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquement

## Représentation en Python ?

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 un dictionnaire :

Table1 = {'ISO': 'AD', 'Name': 'Andorra', 'Capital\_Id': 3041563, … }

On remarque que les clefs ou descripteurs du dictionnaire sont situés dans la première ligne du fichier CSV, les données sont dans les suivantes.

## Lecture d’un fichier CSV en Python

Pour lire un fichier en Python on utilise la fonction **open** qui retourne un objet permettant la manipulation du fichier.

|  |  |
| --- | --- |
| file = open(fichier, "r") |  |
| desc = file.readline() |  |
| list\_desc = desc.rstrip().split(";") |  |
| lignes = file.readlines() |  |
| file.close() |  |

## Liste des méthodes courantes pour manipuler un fichier CSV en Python

* Méthode \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

La méthode .readline() (sans s à la fin) lit une ligne d'un fichier et la renvoie sous forme de chaîne de caractères. À chaque nouvel appel de .readline(), la ligne suivante est renvoyée.

* Méthode \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

La méthode .readlines() agit sur l'objet filin en déplaçant le curseur de lecture du début à la fin du fichier, puis elle renvoie une liste contenant toutes les lignes du fichier (*dans une analogie avec un livre, ceci correspondrait à lire toutes les lignes du livre*).

* Fonction \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ouvre le fichier et permet son utilisation par un objet informatique, il faut fournir le nom et la localisation du fichier et son utilisation en paramètre

* + \_\_\_\_\_  - Read - Default value. Opens a file for reading, error if the file does not exist
  + \_\_\_\_\_  - Append - Opens a file for appending, creates the file if it does not exist
  + \_\_\_\_\_  - Write - Opens a file for writing, creates the file if it does not exist
  + \_\_\_\_\_  - Create - Creates the specified file, returns an error if the file exist
* Méthode \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ : Libère le fichier sur disque
* Méthode \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The rstrip() method removes any trailing characters (characters at the end a string), space is the default trailing character to remove.

* Méthode \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  + The split() method splits a string into a list.
  + You can specify the separator, default separator is any whitespace.

# Relation entre deux tables

countries.csv

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

cities.csv

# Fusion de tables et filtres

Il est possible de fusionner des tables sur un critère cohérent et d’appliquer des filtres ou des tris.

**Par exemple** : le nom de la ville et le nom de son pays du continent Européen

|  |  |  |
| --- | --- | --- |
| **Nom de la ville** | **Nom du pays** | **Continent** |
| Paris | France | EU |
| Luxembourg | Luxembourg | EU |
| Villanova | Italy | EU |
| Moscow | Russia | EU |
| … | … | EU |

## Les tris en pythons

*(à ne pas apprendre par cœur ! Documentation fournis en évaluation)*

Nous verrons plus tard dans l’année plusieurs algorithmes de tris. Pour le moment, découvrons ce que Python nous propose pour trier des types construits.

### La méthode sort() :

* Trie **directement** la liste en place (modifie l'objet).
* Retourne None (elle ne retourne pas une nouvelle liste).
* Syntaxe global :

liste.sort(key=None, reverse=False)

* **key** : Une fonction utilisée pour extraire la clé de tri (facultative).
* **reverse** : True pour un tri décroissant, par défaut False.

nombres = [4, 2, 8, 5, 1]

nombres.sort() # Tri croissant

print(nombres) # [1, 2, 4, 5, 8]

nombres.sort(reverse=True) # Tri décroissant

print(nombres) # [8, 5, 4, 2, 1]

### Fonction sorted()

* Retourne une **nouvelle liste triée** sans modifier l'originale.
* Syntaxe :

sorted(iterable, key=None, reverse=False)

* Exemple :

nombres = [4, 2, 8, 5, 1]

nombres\_tries = sorted(nombres) # Nouvelle liste triée

print(nombres\_tries) # [1, 2, 4, 5, 8]

print(nombres) # [4, 2, 8, 5, 1] (non modifiée)

* La clé de tri (key) permet de personnaliser l'ordre du tri.
* Exemple : Trier une liste de chaînes par longueur.

mots = ["chat", "chien", "aigle", "rat"]

mots.sort(key=len)

print(mots) # ['rat', 'chat', 'chien', 'aigle']

### Trier des dictionnaires

Les dictionnaires en Python ne sont pas directement triables, mais on peut trier :

1. Les **clés**.
2. Les **valeurs**.
3. Les **éléments (clé-valeur)**.

#### Trier par clés

d = {"b": 3, "a": 1, "c": 2}

d\_trie = dict(sorted(d.items()))

print(d\_trie) # {'a': 1, 'b': 3, 'c': 2}

#### Trier par valeurs :

d = {"b": 3, "a": 1, "c": 2}

d\_trie = dict(sorted(d.items(), key=lambda item: item[1]))

print(d\_trie) # {'a': 1, 'c': 2, 'b': 3}

#### Trier par une clé complexe

d = {"b": 3, "a": 1, "c": 2}

d\_trie = dict(sorted(d.items(), key=lambda item: -item[1])) # Tri décroissant par valeurs

print(d\_trie) # {'b': 3, 'c': 2, 'a': 1}

#### Trier une liste de dictionnaires :

Quand une liste contient des dictionnaires, il faut utiliser une clé spécifique pour les trier.

personnes = [

{"nom": "Alice", "age": 30},

{"nom": "Bob", "age": 25},

{"nom": "Charlie", "age": 35}

]

# Tri par âge

personnes.sort(key=lambda personne: personne["age"])

print(personnes)

# [{'nom': 'Bob', 'age': 25}, {'nom': 'Alice', 'age': 30}, {'nom': 'Charlie', 'age': 35}]

\*lambda : Le mot-clé lambda en Python est utilisé pour créer une **fonction anonyme**, c'est-à-dire une fonction sans nom. C'est une manière concise de définir des petites fonctions en une seule ligne, souvent utilisées comme argument dans d'autres fonctions.

|  |  |  |
| --- | --- | --- |
| Pokemon PNG | PNG All | ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; | Pokemon PNG images disponibles pour téléchargement gratuit -  CrazyPNG.com-Crazy PNG images téléchargement gratuit |