I – Table

1. Description

Une table est une organisation de **données structurées**.
Une table contient des colonnes et des lignes.

> Chaque ligne d'une **table**, souvent appelée **enregistrement**, correspond à un objet qui est décrit par des **valeurs** disposées dans chaque colonne. Chaque colonne est donc identifiée par un **descripteur** (ou **champ** ou **attribut**).

Exemple :

Nom	Année	Ville
Andrew	2002	Caen
Steven	2002	Londres
Walter	2007	Paris

Nom, Année et Ville sont les descripteurs.
(Andrew, 2002, Caen) est un enregistrement.

2. Indexation

> La table est **indexée** si chacun de ses enregistrements peut être **identifié de manière unique**.

L'index le plus simple consiste sans doute à numéroter chaque enregistrement.

Exemple :

	Nom	Année	Ville
1	Andrew	2002	Caen
2	Steven	2002	Londres
3	Walter	2007	Paris

Question : pourquoi ne pas choisir le nom comme index pour la table présentée en exemple ?

II – Fichier CSV

1. Description

Rappel SNT :

> Une table peut être enregistrée dans un simple **fichier texte au format CSV** (Comma Separated Values). La première ligne du fichier liste les descripteurs. **Chaque enregistrement est écrit sur une nouvelle ligne**, et les **valeurs sont séparées par un caractère particulier** (originellement une virgule).

2. Importer une table d'un fichier CSV

Il est possible d'importer le contenu d'un fichier CSV en Python en lisant le fichier ligne par ligne. Chaque enregistrement peut alors être affecté dans un tableau (liste Python) ou dans un p-uplet nommé (dictionnaire Python) pour lesquels les clefs sont les descripteurs.
L'ensemble des enregistrements sont eux-mêmes affectés à un tableau global.

> Une table est donc implémentée par un tableau doublement indexé ou par un tableau de p-uplets qui partagent les mêmes descripteurs.

Exemple : La table précédente peut être implémentée par les 2 structures suivantes :

```
>>> table1 = [["Andrew", "2002",  "Caen"],
              ["Steven", "2002", "Londres"],
              ["Walter", "2007", Paris]
             ]

>>> table2 = [{Nom :"Andrew", "Année": "2002", "Ville": "Caen"},
              {Nom :"Steven", "Année": "2002", "Ville": "Londres"},
              {Nom :"Walter", "Année": "2007", "Ville": "Paris"}
             ]
```

Avec le tableau doublement indexé, on accède à la ville de l'enregistrement décrivant Andrew par l'expression `table1[0][2]`, tandis qu'avec le tableau de p-uplets nommés, on y accède par l'expression `table2[0]["Ville"]`.

Remarque : Si l'import du fichier CSV (qui est un fichier texte) se fait sans traitement particulier, toutes les valeurs obtenues sont des chaînes de caractères. On pensera donc à ajouter un traitement pour que les valeurs numériques soient effectivement converties comme telles (`int` ou `float`).

III – Implémentation en Python

1. Implémentation « à la main »

```
f = open('exemple.csv', 'r')
descripteurs = f.readline().strip().split(',')
table1 = [ligne.strip().split(',') for ligne in f.readlines()]
f.close()
```

Remarque : la méthode `strip` permet d'éliminer le caractère `"\n"` de retour à la ligne et la méthode `split` permet de séparer une chaîne de caractères en plusieurs sous-chaînes.

```
f = open('exemple.csv', 'r')
descripteurs = f.readline().strip().split(',')
table2 = []
for ligne in f.readlines():
    ligne = ligne.strip().split(',')
    enregistrement = {}
    for i in range(len(descripteurs)):
        enregistrement[descripteurs[i]] = ligne[i]
    table2.append(enregistrement)
f.close()
```

2. Utilisation d'une librairie annexe

Le **module csv** permet de faire l'import de façon très rapide, comme le montre l'exemple ci-dessous :

```
import csv

############ fonction csv.reader ############
with open('exemple.csv', 'r') as f:
    table1 = list(csv.reader(f, delimiter=','))
# remarque : le 1er sous-tableau est la liste des descripteurs

############ fonction csv.DictReader ############
with open('exemple.csv', 'r') as f:
    table2 = list(csv.DictReader(f, delimiter=','))
```