PREMIERE GENERALE

NUMERIQUE ET SCIENCES INFORMATIQUES

LANGAGE PYTHON : TRAITEMENT DES DONNEES EN TABLES

1 – TABLES ET MATRICES

Un tableau peut être composés de différents types et notamment d'autres tableaux. Un tableau composé de tableaux peut être appelé **table** ou **matrice**. Par exemple, une image peut être représentée au moyen d'une table constituée de lignes et de colonnes où chaque élement représente un pixel.

```
table = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for ligne in table:
    print(ligne)
```

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

Pour construire une table, il possible d'utiliser, à partir d'une table vide, une boucle `for` qui va créer les lignes une à une. Par exemple :

```
table = []
for ligne in range(4):
    ligne = [4*n+i for i in range(1, 5)]
    print(ligne)
    table.append(ligne)
```

```
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, 15, 16]
```

Pour accéder aux différents élèments :

```
table[0]          # accés à la 1ère ligne
```

```
[1, 2, 3, 4]
```

```
table[3][3]          # accés à la dernière ligne
```
```
[13, 14, 15, 16]
```

```
table[3][3]          # accés au dernier élément de la dernière ligne
```
```
16
```

```
table[3][4]          # accés à un élément n'existant pas
```
```
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
IndexError: list index out of range
```

2 – MANIPULATION DES TABLES

2.1 – Présentation du format CSV

Le format **CSV** (pour **C**omma **S**eparated **V**alues, soit en français valeurs séparées par des virgules) est un format informatique permettant de **stocker des tables de données dans un fichier texte**. Les régles du format CSV sont :

- Chaque ligne du fichier correspond à une ligne de la table. Les valeurs de chaque colonne de la table sont séparées par un caractère de séparation, en général une virgule ou un point-virgule.
- Chaque ligne est terminée par un caractère de fin de ligne (line break).
- Toutes les lignes contiennent obligatoirement le même nombre de valeurs (donc le même nombre de caractères de séparation). Les valeurs vides doivent être exprimées par deux caractères de séparation contigus.
- La taille de table est le nombre de lignes multiplié par le nombre de valeurs dans une ligne. La première ligne du fichier peut être utilisée pour exprimer le nom des colonnes (champs).

Par exemple, la table ci-contre, contient la liste (extrait) des prénoms attribués aux enfants nés en France hors Mayotte entre 1900 et 2018 et les effectifs par sexe et département associés à chaque prénom.

sexe	preusuel	annais	dpt	nombre
1	AADIL	1983	84	3
1	AADIL	1992	92	3
1	AAHIL	2016	95	3
1	AARON	1962	75	3
1	AARON	1976	75	3
1	AARON	1982	75	3
1	AARON	1984	75	3
1	AARON	1985	75	5
1	AARON	1989	75	3
1	AARON	1990	69	3
1	AARON	1990	75	4
1	AARON	1990	93	4

Le fichier CSV contenant les données de la table précédente, contient les lignes ci-contre.

```
sexe;preusuel;annais;dpt;nombre
1;AADIL;1983;84;3
1;AADIL;1992;92;3
1;AAHIL;2016;95;3
1;AARON;1962;75;3
1;AARON;1976;75;3
1;AARON;1982;75;3
1;AARON;1984;75;3
1;AARON;1985;75;5
1;AARON;1989;75;3
1;AARON;1990;69;3
1;AARON;1990;75;4
1;AARON;1990;93;4
```

Le format CSV est particulièrement utilisé dans l'**Open Data** (données ouvertes). L'open data, c'est quoi ? Il s'agit de données auxquelles tout le monde peut accéder et que tout le monde peut utiliser et partager. Les gouvernements, les entreprises et les individus peuvent utiliser l'open data afin de créer des avantages sociaux, économiques et environnementaux.

Ces données sont d'origine publique (documents administratifs comme les commandes publiques, données législatives, données cartographiques…) ou privée (papiers de recherche, administration d'entreprises, particuliers et crowdsourcing …).

Un certain nombres de sites internet permettent d'accéder à ces données, parmi lesquels on peut citer : Plateforme ouverte des données publiques françaises, World Bank Open Data, l'INSEE,...

Le fichier **nat2022.csv** qui se trouve dans le répertoire ressource contient les prénoms attribués aux enfants nés en France hors Mayotte entre 1900 et 2022et les effectifs par sexe associés à chaque prénom. La table de données contenue dans ce fichier est constitué de 4 colonnes :

- **sexe** : correspond au sexe de l'enfant - 1 pour les garçons et 2 pour les filles.
- **preusuel** : prénom usuel donné à l'enfant.
- **annais** : année de naissance).
- **nombre** : nombre d'enfants auxquel ce prénom a été donné.

sexe	preusuel	annais	nombre
1	ABDON	1900	4
1	ABEL	1900	428
1	ABRAHAM	1900	19
1	ACHILLE	1900	205
1	ACHILLES	1900	6
1	ADALBERT	1900	7
1	ADAM	1900	20
1	ADELIN	1900	7
1	ADHEMAR	1900	5
1	ADOLPHE	1900	495
1	ADONIS	1900	22

```
sexe;preusuel;annais;nombre
1;ABDON;1900;4
1;ABEL;1900;428
1;ABRAHAM;1900;19
1;ACHILLE;1900;205
1;ACHILLES;1900;6
1;ADALBERT;1900;7
1;ADAM;1900;20
1;ADELIN;1900;7
1;ADHEMAR;1900;5
1;ADOLPHE;1900;495
1;ADONIS;1900;22
```

Exercice n°1

1. **Lancer** un éditeur de texte (Bloc Note par exemple). **Ouvrir** le fichier `nat2022.csv`. **Rechercher** dans les données, votre prénom et votre année de naissance. **Préciser** si cette recherche est facile.
2. **Ouvrir** à nouveau le fichier `nat2022.csv` avec un tableur (Excel par exemple). **Donner** le nombre de lignes qui contenues dans le tableau.
3. **Placer** le curseur sur une case du tableau. **Activer** la fonction « `Filtrer` » (Menu Données).
4. **Appliquer** un filtre sur les prénoms afin de ne garder que votre prénom. **Donner** le nombre d'enfants nés la même année que vous et qui portent votre prénom.
5. **Déterminer** le nombre d'enfants à qui on a donné le même prénom que vous depuis 1900.

2.2 – Importation d'un table

Pour charger un fichier CSV en python et lire les données, il faut utiliser la fonction `open` et la méthode `reader` disponibles dans le module **csv**.

Le code ci-dessous permet d'ouvrir le fichier `nat2022.csv` et de stocker les données dans la table `tab_donnees`.

```
import csv                  # importation de la librairie csv
tab_donnees = []            # Création d'une table vide
with open("nat2022.csv", newline='', encoding="utf-8") as csvfile:   # Ouverture du fichier
    donnees = csv.reader(csvfile, delimiter= ';')      # Lecture des données
    for ligne in donnees:               # Pour chaque ligne de la table
        tab_donnees.append(ligne)       # Stockage de chaque ligne dans la table tab_donnees
```

```
>>> tab_donnees[0:6]
[['sexe', 'preusuel', 'annais', 'nombre'], ['1', '_PRENOMS_RARES', '1900', '1249'], ['1', '_PRENOMS_RARES', '1901', '1342'], ['1', '_PRENOMS_RARES', '1902', '1330'], ['1', '_PRENOMS_RARES', '1903', '1286'], ['1', '_PRENOMS_RARES', '1904', '1430']]
```

Avec le code précédent, les données sont stockées dans un tableau de tableaux. Il n'y a pas de lien direct entre les champs (noms des colonnes, c'est à dire la première ligne de la table tab_donnees) et le reste des données. Pour remédier à cela, la méthode `DictReader` retourne un chaque dictionnaire pour chaque ligne du fichier CSV.

```
import csv                  # importation de la librairie csv
tab_donnees = []            # Création d'une table vide
with open("nat2022.csv", newline='', encoding="utf-8") as csvfile:   # Ouverture du fichier
    donnees = csv.DictReader(csvfile, delimiter= ';')  # Lecture des données
    for ligne in donnees:                  # Pour chaque ligne de la table
        tab_donnees.append(dict(ligne))  # Stockage des dictionnaires dans la table
```

```
>>> tab_donnees[0:6]
[{'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1900', 'nombre': '1249'}, {'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1901', 'nombre': '1342'}, {'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1902', 'nombre': '1330'}, {'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1903', 'nombre': '1286'}, {'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1904', 'nombre': '1430'}, {'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1905', 'nombre': '1472'}]
```

Cette fois-çi on obtient un tableau constitué de dictionnaires correspondant à chaque des lignes du fichier CSV et dont les clés sont les champs du tableau de données. Les données enregistrées dans un fichier CSV sont des **données texte**, donc des données de type `str`. Il peut être souvent nécessaire d'adapter le type de certaines données (entiers, flottants) pour pouvoir les traiter.

Exercice n°2

1. **Compléter** le code ci-dessous afin que les valeurs de la colonne `nombre` soient du type `int` (entier).

```
import csv
tab_donnees = []
with open("nat2022.csv", newline='', encoding="utf-8") as csvfile:
    donnees = csv.DictReader(csvfile, delimiter=';')
    for ligne in donnees:
        tab_donnees.append(dict(ligne))

for ligne in tab_donnees:
    ..........................................................# Convertir les valeurs de la colonne nombre en int
```

2. **Tester** le code.

```
>>> tab_donnees[0:6]
[{'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1900', 'nombre': 1249}, {'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1901', 'nombre': 1342}, {'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1902', 'nombre': 1330}, {'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1903', 'nombre': 1286}, {'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1904', 'nombre': 1430}, {'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '1905', 'nombre': 1472}]
```

2.3 – Recherche dans une table

Il est possible de récupérér des données de la table à partir des différentes clés. Le code ci-dessous permet d'afficher, pour chacune des années depuis 1900, le nombre de filles ayant reçu le prénom "ANNE".

```
for prenom in tab_donnees:
    if prenom['preusuel'] == 'ANNE' and prenom['sexe'] == '2':
        print(prenom['annais'], prenom['nombre'])
```

```
1900 3450
.....
2022 68
```

Le code ci-dessous permet d'afficher le nombre total de fois qu'une fille a reçu le prénom "ANNE" depuis 1900.

```
nb_total = 0
for prenom in tab_donnees:
    if prenom['preusuel'] == 'ANNE' and prenom['sexe'] == '2':
        nb_total = nb_total + prenom['nombre']
```

```
>>> nb_total
```

```
364227
```

Exercice n°3

1. **Editer** un script permettant de connaître le nombre d'enfants nés la même année que vous et qui portent votre prénom.
2. **Editer** un script permettant de connaître le nombre total d'enfants nés depuis 1900 et qui portent votre prénom.
3. **Editer** un script permettant de connaître le prénom le plus donné l'année de votre naissance pour les enfants de votre sexe.

2.4 – Tri d'un table

Pour exploiter les données une table, notamment une table de grande taille, il souvent nécessaire de la trier. La recherche d'un élèment dans une table triée est beaucoup efficace que la recherche d'un élement dans une table quelconque.

Il existe un certain nombre d'algorithmes de tri qui seront vus un peu plus tard dans l'année. Python propose également la fonction **sorted** ou la méthode **sort** qui permettent de trier des tableaux (type `list` Python).

Les lignes de code suivantes permettent de trier les données du fichier **nat2022.csv** en fonction du nombre de fois que le prénom a été donné selon l'**ordre croissant**.

```
def cle_nombre(p) :
    return p['nombre']

tab_donnees.sort(key=cle_nombre)
```

```
>>> tab_donnees[0:6]
```

```
[{'sexe': '1', 'preusuel': 'A', 'annais': '1980', 'nombre': 3}, {'sexe': '1', 'preusuel': 'A',
'annais': '1998', 'nombre': 3}, {'sexe': '1', 'preusuel': 'AADAM', 'annais': '2014', 'nombre':
3}, {'sexe': '1', 'preusuel': 'AADAM', 'annais': '2015', 'nombre': 3}, {'sexe': '1',
'preusuel': 'AADAM', 'annais': '2018', 'nombre': 3}, {'sexe': '1', 'preusuel': 'AADAM',
'annais': '2020', 'nombre': 3}]
```

Les lignes de code suivantes permettent de trier les données du fichier **nat2022.csv** en fonction du nombre de fois que le prénom a été donné selon l'**ordre décroissant**.

```
def cle_nombre(p) :
    return p['nombre']

tab_donnees.sort(key=cle_nombre, reverse=True)
```

```
>>> tab_donnees[0:6]
```

```
[{'sexe': '1', 'preusuel': 'JEAN', 'annais': '1946', 'nombre': 53502}, {'sexe': '2',
'preusuel': 'MARIE', 'annais': '1901', 'nombre': 52150}, {'sexe': '2', 'preusuel': 'MARIE',
'annais': '1902', 'nombre': 51857}, {'sexe': '1', 'preusuel': 'JEAN', 'annais': '1947',
'nombre': 51309}, {'sexe': '2', 'preusuel': 'MARIE', 'annais': '1903', 'nombre': 50424},
{'sexe': '2', 'preusuel': 'MARIE', 'annais': '1904', 'nombre': 50131}]
```

Il est possible de trier la table selon plusieurs critéres. Le code ci-dessous permet de trier les données du fichier `nat2022.csv` selon l'année de naissance puis selon le nombre de fois que le prénom a été donné.

```
def cle_nombre(n) :
    return (n['annais'], n['nombre'])

tab_donnees.sort(key=cle_nombre, reverse=True)
```

```
>>> tab_donnees[0:6]
```

```
[{'sexe': '2', 'preusuel': '_PRENOMS_RARES', 'annais': '2022', 'nombre': 29396}, {'sexe': '1', 'preusuel': '_PRENOMS_RARES', 'annais': '2022', 'nombre': 27296}, {'sexe': '1', 'preusuel': 'GABRIEL', 'annais': '2022', 'nombre': 4889}, {'sexe': '1', 'preusuel': 'LÉO', 'annais': '2022', 'nombre': 4078}, {'sexe': '1', 'preusuel': 'RAPHAËL', 'annais': '2022', 'nombre': 3798}, {'sexe': '1', 'preusuel': 'MAËL', 'annais': '2022', 'nombre': 3571}]
```

Exercice n°4

1. **Compléter** le script ci-dessous permettant de connaître le prénom féminin qui a été donné le plus de fois en une seule année. **Tester**-le.

```
def cle_nombre(p) :
    return p['nombre']
tab_donnees.sort(.....)
i = .....
while tab_donnees[i]['sexe'] == .....:
    i = .....
prenom_max = tab_donnees[i].....
annee_max = tab_donnees[i].....
nb_max = tab_donnees[i].....
```

```
>>> prenom_max, annee_max, nb_max
```

```
('MARIE', '1901', 52150)
```

2. **Compléter** le script ci-dessous permettant de connaître le prénom féminin qui a été donné le moins de fois en une seule année. **Tester**-le

```
def cle_nombre(p) :
    return p['nombre']
tab_donnees.sort(.....)
i = .....
while tab_donnees[i]['sexe'] == .....:
    i = .....
prenom_min = tab_donnees[i].....
annee_min = tab_donnees[i].....
nb_min = tab_donnees[i].....
```

```
>>> prenom_min, annee_min, nb_min
```

```
('AALIA', '2014', 3)
```

3 – EXEMPLE DE TRAITEMENT DE TABLES

Tout au long de l'année, la Mairie de Paris assure la surveillance du patrimoine arboré et recense chacun d'eux dans le fichier `arbres.csv` disponible sur son site internet. La table comporte un certain nombre de données entre autres :

- l'arrondissement de plantation (`ARRONDISSEMENT`) ;
- la classification : nom français (`LIBELLEFRANCAIS`), genre (`GENRE`) et l'espèce (`ESPECE`) ;
- la circonférence (`CIRCONFERENCEENCM`) ;
- la hauteur (`HAUTEUR (m)`) ;
- les coordonnées géographiques (`geo_point_2d`).

Exercice n°5

1. **Editer** un script permettant d'importer les données contenues dans le fichier `arbres.csv`, avec les valeurs de la circonférence (`CIRCONFERENCEENCM`) et de la hauteur (`HAUTEUR (m)`) qui seront converties en nombres flottants.
2. **Tester**-le en affichant la première ligne de la table.

```
>>> tab_donnees[0]
{'IDBASE': '106664.0', 'TYPEEMPLACEMENT': 'Arbre', 'DOMANIALITE': 'Jardin',
'ARRONDISSEMENT': 'PARIS 7E ARRDT', 'COMPLEMENTADRESSE': '', 'NUMERO': '', 'LIEU / ADRESSE':
'JARDIN DU CHAMP DE MARS', 'IDEMPLACEMENT': 'P0090520', 'LIBELLEFRANCAIS': 'Abelia',
'GENRE': 'Abelia', 'ESPECE': 'triflora', 'VARIETEOUCULTIVAR': '', 'CIRCONFERENCEENCM': 75.0,
'HAUTEUR (m)': 6.0, 'STADEDEVELOPPEMENT': 'A', 'REMARQUABLE': '0', 'geo_point_2d':
'48.8551671416,2.3004822832'}
```

Exercice n°6

1. **Modifier** le script précédent afin d'indiquer le nom français (`LIBELLEFRANCAIS`) de l'arbre qui a la plus grande circonférence (`CIRCONFERENCEENCM`) ainsi que l'arrondissement (`ARRONDISSEMENT`) où il est planté.
2. **Tester**-le.

```
>>> nom, circonf_max, arrondissement
('Platane', 784.0, 'BOIS DE BOULOGNE')
```

Exercice n°7

1. **Modifier** le script précédent afin d'indiquer le nom français (`LIBELLEFRANCAIS`) de l'arbre qui a la plus grande hauteur (`HAUTEUR (m)`) ainsi que l'arrondissement (`ARRONDISSEMENT`) où il est planté.
2. **Tester**-le.

```
>>> nom, haut_max, arrondissement
('Erable', 56.0, 'PARIS 18E ARRDT')
```

Certains arbres provoquent de fortes allergies au pollen. Voici la liste de ces arbres : aulne, bouleau, charme, cyprès, frêne, mûrier, noisetier de Byzance, olivier et platane.

On souhaite connaitre le nombre total d'arbres plantés pour chacune de ces espèces ainsi que le nombre total d'arbres provoquant des allergies au pollen.

Exercice n°8

1. **Compléter** le script ci-dessous qui permet de retourner le dictionnaire `arbres_pollen` qui contient les arbres à pollen et pour chaque espèce, le nombre d'arbres plantés à Paris, et qui permet de compter le nombre total d'arbres à pollen.

```
liste_abr = ['Aulne', 'Bouleau', 'Charme', 'Cyprès', 'Frêne', 'Noisetier de Byzance',
             'Mûrier', 'Olivier', 'Platane']
nb_total = ..............
abr_pollen = .................          # dictionnaire vide
for ..........................................................:
    if ligne['LIBELLEFRANCAIS'] ..........................................:
        abr_pollen[ligne['LIBELLEFRANCAIS']]= ..........................................................
        nb_total ..............................
```

2. **Tester**-le.

```
>>> nb_total
```

```
60293
```

```
>>> abr_pollen
```

```
{'Aulne': 713, 'Bouleau': 2425, 'Charme': 3505, 'Cyprès': 1254, 'Frêne': 5159, 'Mûrier':
    918, 'Noisetier de Byzance': 3407, 'Olivier': 85, 'Platane': 42827}
```

3. **Préciser** quelle espèce d'arbre à pollen est la plus présente à Paris.

Exercice n°9

1. **Modifier** le script précédent afin de retourner le dictionnaire `arrond_pollen` qui contient les arrondissements et pour chacun, le nombre d'arbres à pollen plantés.

2. **Tester**-le.

```
>>> arrond_pollen
```

```
{'PARIS 12E ARRDT': 4740, 'PARIS 19E ARRDT': 3933, 'PARIS 15E ARRDT': 3802, 'PARIS 13E
    ARRDT': 3985, 'PARIS 20E ARRDT': 3448, 'PARIS 7E ARRDT': 4278, 'PARIS 17E ARRDT': 3929,
    'PARIS 16E ARRDT': 4910, 'PARIS 18E ARRDT': 3003, 'PARIS 14E ARRDT': 3447, 'PARIS 11E
    ARRDT': 2170, 'PARIS 5E ARRDT': 1121, 'SEINE-SAINT-DENIS': 2575, 'HAUTS-DE-SEINE': 1981,
    'VAL-DE-MARNE': 2338, 'BOIS DE VINCENNES': 1981, 'PARIS 3E ARRDT': 455, 'PARIS 4E ARRDT':
    992, 'PARIS 1ER ARRDT': 627, 'PARIS 8E ARRDT': 2231, 'PARIS 9E ARRDT': 493, 'PARIS 10E
    ARRDT': 1589, 'PARIS 2E ARRDT': 155, 'PARIS 6E ARRDT': 801, 'BOIS DE BOULOGNE': 1309}
```

3. **Trier** le dictionnaire `arrond_pollen` .à l'aide de la ligne de code suivante :

```
>>> sorted(arrond_pollen.items(), key=lambda t:t[1])
```

4. **Préciser** quel arrondissement peut-on conseiller aux personnes souffrant d'allergie au pollen.