Important : Télécharger ce fichier (TP_5.pdf) dans votre répertoire IP2/TP05.

Exercice 1 :

L'**INSEE** (Institut National de la Statistique et des Études Économiques) est une source ouverte de données ou " open data ". Il propose en particulier : le fichier des prénoms qui contient des données sur les prénoms attribués aux enfants nés en France entre 1900 et 2020.

1) Ouvrir l'adresse : https://www.insee.fr/fr/statistiques/2540004#consulter

Télécharger le zip du fichier CSV correspondant à la France hors Mayotte.

Placer le zip dans votre répertoire de travail (IP2/TP05).

Dézipper en double cliquant sur le zip le fichier.

Déplacer le fichier contenu dans le répertoire qui vient d'apparaître dans votre répertoire de travail (IP2/TP05).

Important : AVANT DE CONTINUER VOUS DEVEZ AVOIR le FICHIER TP_5.pdf et le fichier nat2021.csv dans votre répertoire IP2/TP05 .

2) Tenter l'ouverture du fichier avec LibreOffice. Vous devriez obtenir ce genre de visualisation. Soyez patient(e)s, le fichier est énorme.

Le premier fichier national comporte 686 593 enregistrements et quatre variables décrites ci-après.

Ce fichier est trié selon les variables SEXE, PREUSUEL, ANNAIS.

- SEXE : 1 pour masculin - 2 pour féminin.
- PRESUEL : prénom usuel.
- ANNAIS : contraction de ANnée de NAISsance. 1900 à 2021
- NOMBRE : la fréquence du prénom dans l'année.

	A	B	C	D
1	sexe	preusuel	annais	nombre
2	1	_PRENOMS_RARES	1900	1249
3	1	_PRENOMS_RARES	1901	1342
4	1	_PRENOMS_RARES	1902	1330
5	1	_PRENOMS_RARES	1903	1286
6	1	_PRENOMS_RARES	1904	1430
7	1	_PRENOMS_RARES	1905	1472
8	1	_PRENOMS_RARES	1906	1451
9	1	_PRENOMS_RARES	1907	1514

FIGURE 1 – Début du Fichier, l'ensemble des prénoms très rares sont regroupés par année.

Toujours fermer le fichier après l'avoir consulté pour éviter les conflits d'accès avec Python par la suite.

1	GERIC	1982	4
1	GERIC	1985	3
1	GERIC	1992	3
1	GERIC	1994	4
1	GERIC	XXXX	20
1	GERLANDO	1959	3
1	GERLANDO	1960	3
1	GERLANDO	XXXX	29
1	GERMAIN	1900	510
1	GERMAIN	1901	549
1	GERMAIN	1902	637
1	GERMAIN	1903	611

FIGURE 2 – Une année XXXX sert de séparation dans la liste des prénoms. Ces lignes seront ignorées par la suite. Cet extrait montre qu'il y a eu 637 garçons (code 1) appelés Germain en 1902.

3) Ouvrez le fichier avec le logiciel gedit. Vous devriez obtenir le visuel suivant. Gedit vous permet de voir le fichier avec les séparateurs de données, à savoir ici les " ;".

Avec la souris clic droit sur le fichier csv, choisir ouvrir avec ...puis gedit

```
 1 sexe;preusuel;annais;nombre
 2 1;_PRENOMS_RARES;1900;1249
 3 1;_PRENOMS_RARES;1901;1342
 4 1;_PRENOMS_RARES;1902;1330
 5 1;_PRENOMS_RARES;1903;1286
 6 1;_PRENOMS_RARES;1904;1430
 7 1;_PRENOMS_RARES;1905;1472
 8 1;_PRENOMS_RARES;1906;1451
 9 1;_PRENOMS_RARES;1907;1514
10 1;_PRENOMS_RARES;1908;1509
11 1;_PRENOMS_RARES;1909;1526
```

FIGURE 3 – Le logiciel gedit permet de voir le contenu brut du fichier. Sur chaque ligne les données sont séparées par un " ;" et non un virgule. Il sera tout de même possible de le charger avec Python.

Toujours fermer le fichier après l'avoir consulté pour éviter le conflits d'accès avec Python par la suite.

Avant de passer à la suite, fermer tous les outils de visualisation du fichier csv. Par la suite vous pourrez consulter à nouveau ces fichiers avec libre office ou gedit, si vous estimez que c'est nécessaire.

4) Chargement des données. Dans le CM3 vous avez vu comment charger un fichier csv. Par défaut le fichier csv a pour séparateur de données, la virgule. Ici pour le fichier de l'INSEE, le séparateur ou délimiteur est le point-virgule " ;". Dans la fonction vue en CM3 vous devez donc l'indiquer en ajoutant un paramètre (" delimiter=' ;' ") lors de l'appel de csv.reader.

Travail à faire :

1) Créer dans votre répertoire TP05 le fichier fonctions_fichier_csv.py

2) Convertir le code du CM pour écrire la fonction chargeFichierCsv dans le fichier fonctions_fichier_csv

3) tester cette fonction. .

Pour tester la fonction, vous pouvez regarder la longueur du fichier, visualiser les premières lignes ou les dernières lignes et comparer avec les résultats des questions précédentes. Le code suivant que vous devez compléter vous propose des tests dans l'auto test. N'hésitez pas à en rajouter pour vous convaincre que les données sont correctement chargées.

Le code ci dessous est la trame du fichier **fonctions_fichier_csv.py** :

AJOUTER VOS LIGNES À LA PLACE DES LIGNES COMMENCANT PAR ##

fonctions_fichiers_csv.py

```python
import csv

def chargeFichierCsv(nom : str) -> list[list[str]] :
        """ Lit un fichier au format csv séparé par des ;"""
        ## reporter en adaptant le code du CM ICI
        ## bien  modifier par rapport au cours   delimiter=";"
    reader = csv.reader(fichier,delimiter=';') # lecture

def ecritFichierResultat(nomFichierSortie : str, data :list[list[str]] ) :
        """ecrit un fichier au format csv"""
    pass  # ne pas modifier cette fonction.

if __name__=="__main__" :
    data=chargeFichierCsv("nat2021.csv")
    assert (data[500]==['1', 'ABD', '1954', '20'])
    assert(len(data)==686539)
    print(len(data))
    print(data[500])
    # AJOUTER VOS TESTS ICI
```

5) Votre fonction de la question 4 va être importée dans un autre code.Vous allez importer le module écrit précédemment. Vous allez tester que l'importation se passe bien.

Travail à faire :

- **DANS VOTRE REPERTOIRE DE TRAVAIL IP2/TP05 Créer le fichier EX_1_prenoms.py .**
- **VERIFIER QUE VOUS AVEZ LE FICHIER CSV, et les 2 FICHIERS EX_1_prenoms.py et fonction_fichiers_csv.py dans le répertoire de travail IP2/TP05.**
- Copier et coller les lignes de code suivantes dans **EX_1_prenoms.py**
- Compléter ce code avec vos tests de chargement, lecture en complément de ceux déja proposés.

EX_1_prenoms.py

```python
import fonctions_fichiers_csv  as fic

def testImport(nom_fichier : str )->list[list[str]] :
    """Cette fonction permet de vérifier l'import
    correct du module et de son bon fonctionnement"""
    data : list[str] =fic.chargeFichierCsv(nom_fichier)

    return data

nom_fichier="nat2021.csv"
data : list[list[str]] = testImport(nom_fichier)
assert (data[500]==['1', 'ABD', '1954', '20'])
assert(len(data)==686539)
print(len(data))
print(data[500])
# AJOUTER VOS TESTS ICI (vérifier une autre ligne de data par exemple)
```

6) Dans votre fichier EX_1_prenoms.py , vous allez :

- commenter (ajouter un #) devant les lignes nom_ficher=... (lignes 16 à 21 du code ci -dessous)
- ajouter les lignes de codes (23 à 28) ci-dessous dans votre fichier EX_1_prenoms.py.
- compléter (et tester !!) la fonction **extrait_data** (écrire votre code dans les lignes 11 et suivantes). Cette fonction lit la liste de données data et renvoie quatre listes de même longueur.

Voir les lignes de code à rajouter dans votre fichier EX_1_prenoms.py sur la page suivante.

ligne de code à ajouter

```python
def extrait_data(data:list[list[str]]) :
    """Cette fonction extrait les données
    et renvoie 4 listes de même longueur"""
    sexe : list =[]
    prenom :list =[]
    annees :list =[]
    nb :list =[str]
    for k in range(len(data)):
        ligne=data[k]
        sexe.append(ligne[0 ])
        #
        #vos lignes de codes ici
        #
    return sexe, prenom, annees, nb

#nom_fichier="nat2021.csv"
#data : list[list[str]] = testImport(nom_fichier)
#assert (data[500]==['1', 'ABD', '1954', '20'])
#assert(len(data)==686539)
#print(len(data))
#print(data[500])

nom_fichier="nat2021.csv"
data : list[str] =fic.chargeFichierCsv(nom_fichier)
sexe, prenom, annees, nb = extrait_data(data)
assert(prenom[500] =='ABD')
assert(annees[500]=='1954')
assert(nb[500] = '20')
```

7) Recopier , éxécuter, tester dans votre fichier EX_1_prenoms.py le code suivant. ATTENTION : pour des raisons de mise en page sur ce document, des retours à la ligne ont été ajoutés (ligne 1 à 6) ainsi que dans l'appel (à partir de la ligne 24). A vous d'éliminer ces erreurs éventuelles.

code à rajouter

```
1  def extraitDonneesPourPrenom(PrenomCherche : str ,
2                               Genre: int ,
3                               sexe:list[str],
4                                prenom :list[str],
5                                 annees:list[str],
6                                  nb:list[str]) :
7      """Affiche le nombre de fois où un prénom est donnée en fonction
8      de l'année de naissance
9      Retourne la liste des années et des effectifs correspondants
10     """
11     ans       : list[int]=[]    # années où le prénom a été donné
12     effectifs : list[int]=[]    # nombre du prénom cherché dans l'année.
13
14     for k in range(len(prenom)):
15         if str(prenom[k])== str(PrenomCherche) and Genre==int(sexe[k]) :
16             if annees[k]!='XXXX':
17                 ans.append(int( annees[k] ))
18                 effectifs.append(int( nb[k] ))
19
20     return ans,effectifs
21
22 PrenomCherche='CAMILLE'  # prénom  épicène, prévoir les 2 cas de recherche.
23 Genre=2
24 ans2,effectifs2 = extraitDonneesPourPrenom(PrenomCherche, Genre,
25                                            sexe, prenom, annees, nb)
26 Genre=1
27 ans1,effectifs1 = extraitDonneesPourPrenom(PrenomCherche, Genre,
28                                            sexe, prenom, annees, nb)
```

8) Maintenant que vous savez extraire les données pour un prénom vous allez effectuer le tracé de la courbe donnant le nombre du prénom cherché par ans. Un exemple est fourni ci-après.

travail à faire : Dans votre fichier EX_1_prenoms.py , ajouter la fonction de tracé graphique en utilisant les commandes matplotlib. Pour les prénoms épicènes, représenter les 2 courbes, sinon une seule. A vous de trouver le critère numérique permettant de voir qu'un prénom est épicène.

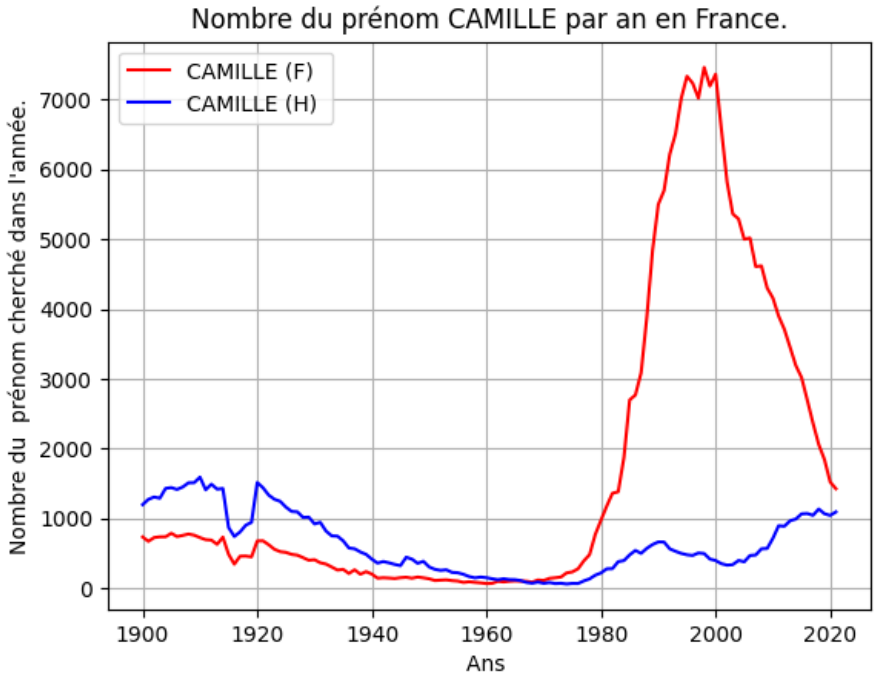

FIGURE 4 – Représentation graphique du nombre de prénom par an.

9) Effectuez d'autres recherches de prénoms de votre choix. Si vous souhaitez conserver une trace de votre travail vous pouvez utiliser l'icône disque sur le graphique matplotlib (en bas à droite) pour obtenir un fichier *.png.

Exercice 2 :

Travail à faire :

- Créer un nouveau, fichier nommé EX2.py.
- Copier et coller dans EX2.py le code suivant.
- Ajouter vos lignes de code après chaque question.
- Faire pour chaque question des affichages explicites, en affichant le numéro de la question et les valeurs obtenues. Rédiger de façon à ce que l'on lise clairement ce que vous avez obtenu.

EX2.py

```
import numpy as np
# 1. Créer le tableau (array) x=[ 1 2 3 4 5 ]

print(f"1)  x = {x}")   # affichage explicite du résultat.
print()   # pour laisser une ligne vide.(+ lisible.)
# 2. Afficher le type de x puis sa longueur.

print...    # affichage du résultat.
print()

# 3. Extraire le premier élément, puis en faire de même avec le dernier.
# Afficher les résultats dans une phrase explicite.

# 4. Extraire les trois premiers éléments et les stocker dans un tableau
# que  l'on nommera a.
# Afficher les résultats dans une phrase explicite.


# 5. Extraire les 1er, 2e et 5e éléments du tableau (array) (attention aux positions)
#  les stocker dans un tableau que l'on nommera b.
# Afficher les résultats dans une phrase explicite.
```

EX2.py - SUITE

```
# 6. Additionner le nombre 10 au tableau x,
#  puis multiplier le résultat par 2.
# Afficher les résultats dans une phrase explicite.

# 7. Effectuer l'addition de a et b, commenter le résultat.
# Afficher les résultats dans une phrase explicite.

# 8. Effectuer l'addition suivante : x+a ; commenter le résultat,
# puis regarder le résultat de a+x.


# 9. Multiplier le tableau  par le scalaire c que l'on fixera à 2.
# Afficher les résultats dans une phrase explicite.

# 10. Effectuer la multiplication de a et b ; commenter le résultat.
# Afficher les résultats dans une phrase explicite.

# 11. Effectier la multiplication suivante : x*a ; commenter les résultats.
# Afficher les résultats dans une phrase explicite.

# 12. Récupérer les positions des multiples de 2 et les stocker
# dans un tableau  que l'on nommera ind,
# puis conserver uniquement les multiples de 2 de x dans un tableau (array)
# que l'on nommera mult_2.
# Afficher les résultats dans une phrase explicite.

# 13. Afficher les éléments de x qui sont multiples de 3 et multiples de 2.
# Afficher les résultats dans une phrase explicite.
```

EX2.py - SUITE

```
# 14. Afficher les éléments de x qui sont multiples de 3 ou multiples de 2.
# Afficher les résultats dans une phrase explicite.


# 15. Calculer la somme des éléments de x.
# Afficher les résultats dans une phrase explicite.

# 16. Remplacer le premier élément de x par un 4.
# Afficher les résultats dans une phrase explicite.

# 17. Remplacer le premier élément de x par la valeur nan.
# Re-définir le tableau avec dtype=float,
#  puis calculer la somme des éléments de x.
# Afficher les résultats dans une phrase explicite.

# 18 Supprimer le tableau  x. Le vérifier avec try-except.
# Afficher que le tableau n'existe plus dans une phrase explicite.
```