RAPPORT SAE 15
TRAITER DES DONNÉES

Initiation aux traitement des données via le langage Python 3

PICHOT Owen

01/2023

Réseaux & Télécoms – Groupe 122

1. SOMMAIRE

2. INTRODUCTION

a. But de la SAE

Pour ce projet, j'ai dû mettre en place un système de tirage loto, soit 5 chiffres tirés au hasard compris entre 1 et 45 et permettre ensuite de manipuler ces données par exemple en les triant, les affichants sous forme de graphique pour analyse plus simple visuellement ou bien encore de rechercher un certain nombre dans l'un desdits tirage.

Cette SAE permet notamment de s'entraîner premièrement sur la gestion et le traitement de grandes quantités de données (relativement parlant par rapport à mon niveau actuel) et secondement sur le langage de programmation PYTHON et les algorithmes en général.

Ce projet permet également une amélioration de nos compétences organisationnelles sur la gestion du temps, d'un projet assez long et de la rédaction d'un rapport détaillé et clair permettant la synthèse globale du travail effectué.

b. Résumé de mon projet

Pour mon projet, j'ai décidé de créer un programme complet sans fin à proprement parler en créant une boucle infinie de celui-ci. On peut de ce fait générer plusieurs tirages et les enregistrer sous plusieurs formats, créer une représentation graphique des résultats obtenus et charger d'anciens traitements depuis un fichier tiers afin d'en manipuler son contenu.

Même si je n'ai jamais réellement programmé auparavant un projet de cette envergure en Python, j'ai pu m'aider de mes expériences passées avec le Javascript et le langage C notamment dans la structuration globale de mon programme et les intéractions avec l'utilisateur.

Les principales contributions à ce programme ont été les cours dispensés à l'école par Mr Bennis Ismail mais également et surtout grâce aux 'écoles en ligne' du type CodeCademy et Khan Academy proposant tous deux de nombreuses ressources sur la programmation en général.

La rédaction de mon programme n'aura pas eu recours à des outils automatisés en ligne du type ChatGPT et autres, le but du projet étant d'améliorer nos compétences en programmation, l'utilisation de tels outils irait à l'encontre du but.

c. Présentation du programme global (loop)

Avant d'entrer en détail dans chaque partie du code, j'aimerais commencer par présenter dans cette partie l'organisation globale de la boucle principale et des intéractions possibles avec celui-ci en fonction des choix sélectionnés par un utilisateur.

NB : Organigramme créé grâce à la bibliothèque Lucid Charts.

L'organisation de la boucle est comme ci :

Globalement, on retourne à chaque fois au menu principal (avec des options en plus si un tirage a déjà été effectué).

On y retrouve alors la possibilité de soit analyser / modifier le dernier tirage effectué ou bien en charger un nouveau pré-existant afin de le manipuler.

Des sélections de filtres pour les nouveaux tirages sont disponibles comme la gestion de la seed utilisée, le tri à effectuer (ou pas), le graph à générer (ou pas) et la recherche dichotomique.

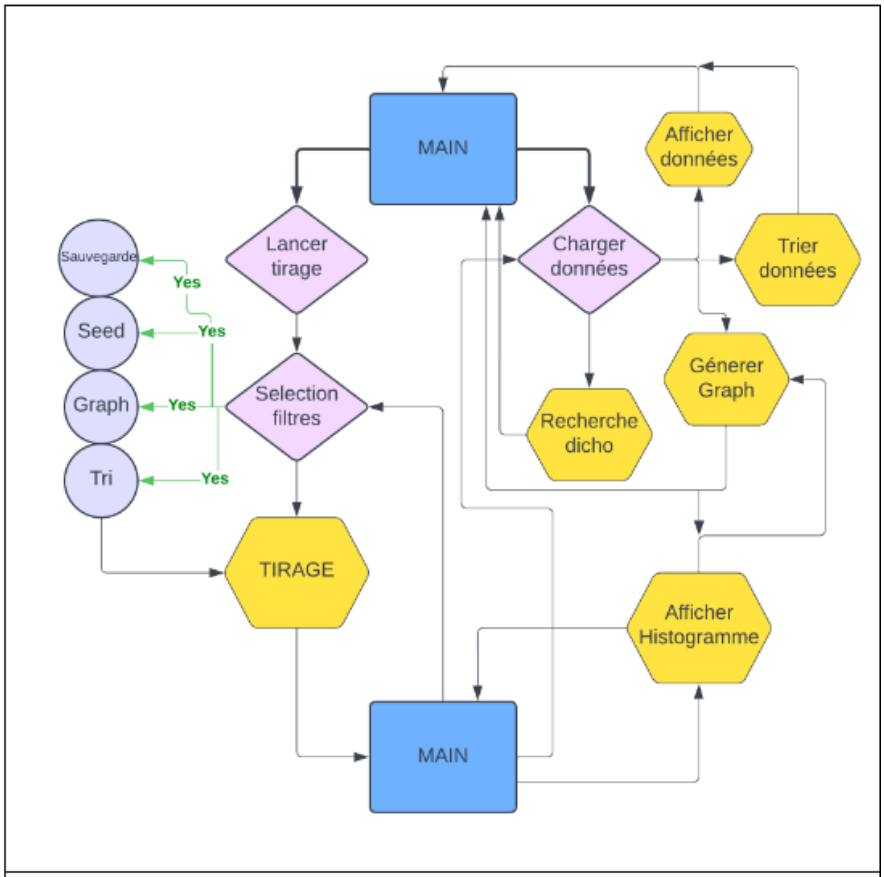


Figure 1 - Graphique de la boucle principale de main.py

d. Organisation des fichiers et dépendances utilisées

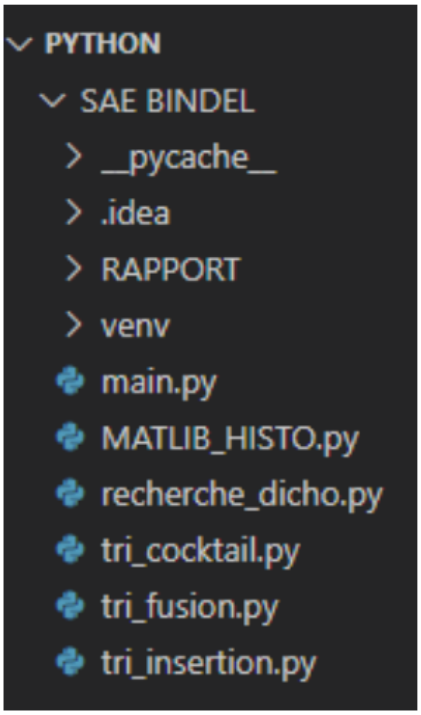


Figure 2 - Arborescence de fichiers du projet

Le programme principal (main.py) gère le programme dans sa globalité, celui fait appel aux sous-fichiers respectifs pour les tri, la recherche dichotomique et la génération de l'histogramme.

Le sous-dossier '/venv', quant à lui, contient les dépendances requises au bon fonctionnement du programme, notamment :

- Numpy
- Matplotlib
- json / csv / pickle
- os

Il est à noter que lors d'un test sur une autre machine, il peut être nécessaire d'installer à la main les dépendances.

Il suffit dans ce cas d'effectuer les commandes suivantes dans la console de votre IDE :

- pip install numpy
- pip install matplotlib

3. LES ALGORITHMES ET LE CODE

a. Recherche dichotomique

L'algorithme de recherche dichotomique permet de déterminer la position d'un nombre dans un tableau.

La version itérative est plutôt simple et consiste simplement à rechercher le tableau valeur par valeur jusqu'à tomber sur celle que l'on recherche, auquel cas on renvoi la position à laquelle on se situe actuellement, le tableau n'a donc pas nécessairement besoin d'être trié car l'algorithme ne repose pas sur les valeurs en soi contenues dans le tableau mais uniquement sur la position à laquelle nous sommes.

La version récursive cependant repose entièrement sur le fait que le tableau soit trié. Le principe est de déterminer le milieu du tableau (valeur obligatoirement entière donc) puis on regarde si la valeur que l'on cherche est plus petite que la valeur de la case.Si tel est le cas alors on continue a chercher sur la moitié à droite (forcément plus grande) sinon, on cherche sur la moitié gauche.

Complexité (pour un nombre de comparaison n) : T(n)=1+T(n/2)

ALGORITHME VERSION ITÉRATIVE :

Données :

- *Nbr*, Nombre à rechercher dans la suite.
- *T*, Tableau dans lequel est effectué la recherche

```
Fonction Dicho_ité (Nbr, T)
        position ← 0
        pour i ← 0 à taille(T) faire
                si i > taille(T) alors
                        retourner "Nombre non présent dans le tableau"
                fsi
                sinon si Nbr = T[i] alors
                        retourner position
                fsi
                sinon
                        position ← position + 1
                fsi
        fin pour
```

ALGORITHME VERSION RÉCURSIVE:

N.B : Signe // = division entière, produit le quotient de deux arguments (toujours entier).

Données :

- *Nbr*, Nombre à rechercher dans la suite
- *T*, Tableau dans lequel est effectué la recherche
- *D*, Position début du tableau
- *F*, Position fin du tableau

Fonction Dicho_récu (*Nbr, T, D, F*)
    si *D* > *F* **alors**
        retourner “Nombre non présent dans le tableau”
    fsi
    milieu ← (*D* + *F*) // 2
    si *T[milieu]* = *Nbr* **alors**
        retourner *milieu*
    fsi
    sinon si *T[milieu]* > *Nbr* **alors**
        retourner Dicho_récu (*Nbr*, *T*, *D*, (*milieu* - 1))
    fsi
    sinon
        retourner Dicho_récu (*Nbr*, *T*, (*milieu* + 1), *F*)
    fsi

CODE VERSION ITÉRATIVE :

```python
def dicho_ité(Nbr, ARR):
    position = 0
    for i in range(len(ARR)):
        if i > len(ARR):
            return (f"Nombre {Nbr} non présent dans la ligne")
        elif Nbr == ARR[i]:
            return (f"Le nombre est présent à la position {position + 1} du tableau")
        else:
            position += 1
```

Figure 3 - Présentation du code Python de l'algorithme de recherche dichotomique, version itérative

CODE VERSION RECURSIVE:

```python
def dichotomie_recursive(nombre, liste, debut, fin):
    if debut > fin:
        return "L'élément n'existe pas dans la liste"
    milieu = (debut + fin) // 2
    if liste[milieu] == nombre:
        return milieu
    elif liste[milieu] > nombre:
        return dichotomie_recursive(nombre, liste, debut, milieu-1)
    else:
        return dichotomie_recursive(nombre, liste, milieu+1, fin)
```

Figure 4 - Présentation du code Python de l'algorithme de recherche dichotomique, version récursive

b. Tri Cocktail

L'algorithme de tri cocktail fonctionne en se 'promenant' de gauche à droite puis de droite à gauche dans un tableau jusqu'à ce que chaque valeur soit bien triée dans un ordre croissant.

Lorsqu'il commence à la position 0 d'un tableau il va continuellement prendre la valeur la plus grande qu'il encontre et la déplacer d'une position vers la droite jusqu'à rencontrer une valeur plus grande que celle qu'il déplace, auquel cas, il laissera la valeur actuelle à la position de la nouvelle et déplacera à la place la nouvelle valeur plus grande et ainsi de suite.

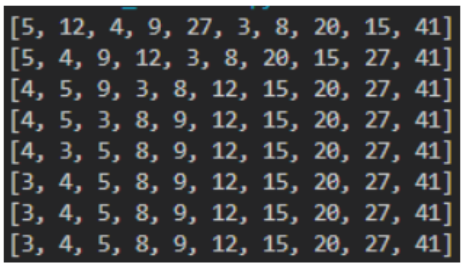


Figure 5 - Exemple de l'effet du tri cocktail itération après itération sur une suite non-trié

Une fois arrivé à la fin du tableau, il effectue le chemin inverse en prenant cette fois à chaque fois la valeur la plus petite qu'il rencontre toujours en la remplaçant si il croise une valeur plus petite que celle qu'il déplace.

Complexité (optimisé) = n^1

CODE DU TRI COCKTAIL :

```python
def tri_cocktail(array):
    moved = True
    while moved:
        moved = False

        for i in range(0, len(array)-1):
            if array[i] > array[i + 1]:
                array[i], array[i + 1] = array[i + 1], array[i]
                moved = True
        for i in range(len(array)-1, 0):
            if array[i] > array[i + 1]:
                array[i], array[i + 1] = array[i + 1], array[i]
                moved = True
        #print(array)
    return array
```

Figure 6 - Présentation du code Python de l'algorithme de tri cocktail

ALGORITHME DU TRI COCKTAIL:

Données :

- *T*, Tableau à trier

```
Fonction tri_cocktail (T)
    déplacé ← Vrai
    tant que déplacé faire
        déplacé ← Faux
        pour i ← 0 à (taille(T) - 1) faire
            si T[i] > T[i + 1] alors
                T[i], T[i + 1] ← T[i + 1], T[i]
                déplacé ← Vrai
            fsi
        fin pour
        pour i ← (taille(T) - 1) à 0 faire
            si T[i] > T[i + 1] alors
                T[i], T[i + 1] ← T[i + 1], T[i]
                déplacé ← Vrai
            fsi
        fin pour
    fin tant
    retourner T
```

c. Tri par insertion

L'algorithme de tri par insertion est certes, moins efficace que d'autres comme le tri fusion sur les grandes séquences de données à trier mais est cependant très efficace sur de plus petites suites de nombres ou des suites avec des nombres déjà presque triés.

Cet algorithme fonctionne de manière plutôt simple en partant du départ du tableau et avançant au fur et à mesure sur la droite de celui-ci.

Il va ensuite pour chaque nombre qu'il rencontre plus petit que celui sur lequel il était le déplacer à gauche d'une position autant de fois que nécessaire jusqu'à ce que le prochain chiffre à sa gauche soit enfin plus petit que celui qu'il déplace.

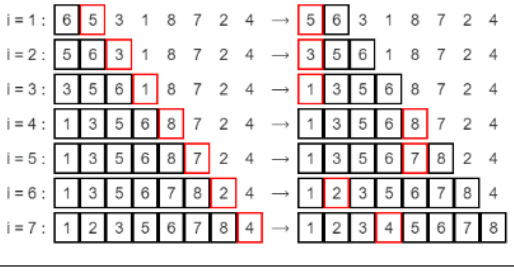


Figure 7 - Exemple schématique de l'exécution d'un algorithme de tri par insertion

Une fois arrivé à la fin du tableau, la suite est donc triée.

Complexité (optimisé) = n^2

Données :

- *T*, Tableau à trier

Fonction tri_insertion (*T*)

```
pour i ← 1 à taille(T) faire
    x ← T[i]
    j ← i
    tant que j > 0 ET T[j - 1] > x faire
        T[j] ← T[j -1]
        j ← j - 1
    T[j] ← x
fin pour
retourner T
```

CODE DU TRI PAR INSERTION:

```python
def tri_insertion(array):
    for i in range(1, len(array)):
        x = array[i]
        j = i
        while j > 0 and array[j - 1] > x:
            array[j] = array[j - 1]
            j = j - 1
        array[j] = x
    return array
```

Figure 8 - Présentation du code Python de l'algorithme de tri par insertion

d. Tri Fusion

L'algorithme de tri fusion porte ce nom car il repose sur le fait de diviser encore et encore des tableaux afin de les trier chacun de leur côté puis de rassembler ceux-ci.

Son utilisation pour rentrer un peu plus dans le détail fonctionne comme ci, on divise le tableau en deux puis on divise de nouveau chaque tableau en deux etc etc jusqu'à arriver à des tableaux de 2 valeurs chacunes.

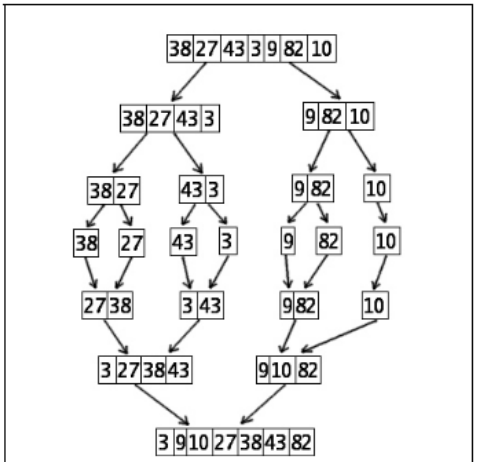


Figure 9 - Exemple schématique de l'exécution d'un algorithme de tri fusion

Une fois cette situation en place, il n'y a plus qu'à mettre les deux valeurs de chaque tableau dans l'ordre croissant puis de rassembler deux fois deux tableaux en prenant les premières valeurs de chaque et les mettre dans l'ordre croissant et ainsi de suite, jusqu'à finalement revenir aux deux derniers tableaux que l'on rassemble afin d'obtenir le résultat final trié.

Complexité : La complexité est logarithmique dû à la division des tableaux en deux continuellement. La complexité est donc de : n log n

N.B : Signe // = division entière, produit le quotient de deux arguments (toujours entier).

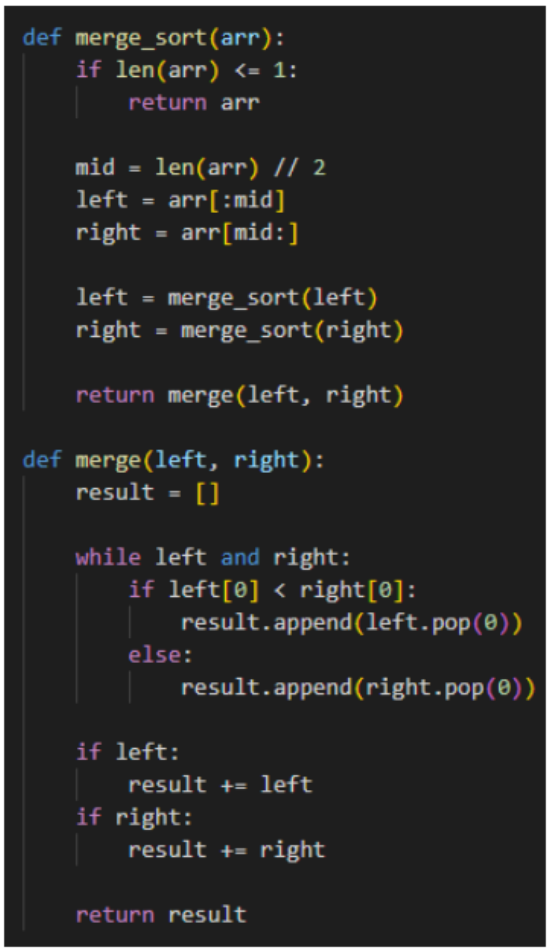
ALGORITHME DU TRI FUSION:

Données :

- *T*, Tableau à trier
- *gauche*, Partie gauche d'un tableau
- *droite*, Partie droite d'un tableau
- *X*, Tableau vide à l'origine

```
Fonction tri_fusion (T)
    si taille(T) <= 1 alors
        retourner T
    fsi
    milieu ← taille(T) // 2
    gauche ← T[avant - milieu]
    droite ← T[après - milieu]

    gauche ← tri_fusion (gauche)
    droite ← tri_fusion (droite)

    retourner fusion (gauche, droite)



Fonction fusion (gauche, droite)
    X ← []
    tant que taille(gauche) > 0 ET taille(droite) > 0 faire
        si gauche[0] < droite[0] alors
            ajouter à X gauche[0]
            retirer[dernière position] gauche
        sinon
            ajouter à X droite[0]
            retirer[dernière position] droite
        fsi
    fin tant
    si taille(gauche) > 0 alors
        ajouter gauche à la fin de X
    fsi
    si taille(droite) > 0 alors
        ajouter droite à la fin de X
    fsi

    retourner X
```

CODE PYTHON DU TRI FUSION:



```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left = arr[:mid]
    right = arr[mid:]

    left = merge_sort(left)
    right = merge_sort(right)

    return merge(left, right)

def merge(left, right):
    result = []

    while left and right:
        if left[0] < right[0]:
            result.append(left.pop(0))
        else:
            result.append(right.pop(0))

    if left:
        result += left
    if right:
        result += right

    return result
```

Figure 10 - Présentation du code Python de l'algorithme de tri fusion

NOTE : Lors de l'implémentation de cet algorithme dans mon code global (main.py), une erreur se produisait notamment lors de l'envoi de données provenant d'un fichier Json.

Il y a donc un ajout d'une troisième fonction (recept_Data) permettant de récupérer le tableau à trier et de le passer dans un format de liste Python très standard afin d'éviter tout problème.

e. Histogramme

Un histogramme est une représentation souvent graphique permettant de visualiser certaines données. Dans le cas du tirage du loto, l'histogramme sera créé à partir du nombre de fois où chaque numéro entre 1 et 45 est tiré.

Voici ci dessous l'algorithme que j'ai utilisé pour créer mon tableau utilisé lors de la création de l'histogramme, il est à noter que la fonction ci-dessous est à utiliser après chaque tirage de 5 nombres (ou plus) et non en lui donnant le tableau final global contenant l'entièreté des tirages.

Le système est simple, je génère d'abord un tableau contenant 45 valeurs toutes mises au départ à zéro. Je prend ensuite la valeur à compter dans l'histogramme et ajoute 1 dans le tableau T à la position [n - 1].

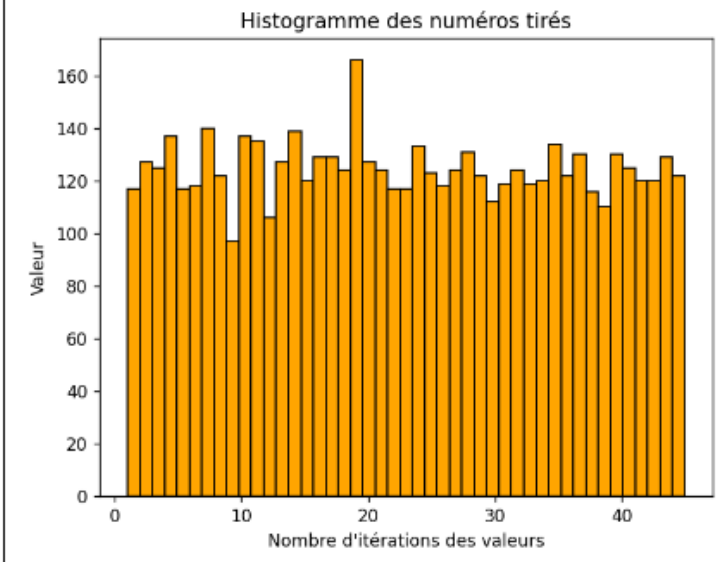


Figure 11 - Exemple du résultat d'un histogramme généré grâce au tableau créé par l'algorithme

ALGORITHME DE GÉNÉRATION DU TABLEAU POUR L'HISTOGRAMME:

Données :

- *T*, Tableau contenant 5 valeurs entre 1 et 45.

histo_count ← []

pour *i* ← 0 **à** 45 **alors**

histo_count[i] ← 0

fin pour

Fonction add_Count *(T)*

pour *x* ← 0 **à** 5 **alors**

histo_count[T[x] - 1] += 1

fin pour

CODE DE GÉNÉRATION DU TABLEAU POUR L'HISTOGRAMME:

```
histo_count = []

for i in range(1,46):
    histo_count.append(0)

# print(histo_count[0])# == 1
# print(histo_count[44])# == 45

def add_Count(tirage):
    for x in range(5):
        histo_count[tirage[x]-1] += 1
```

Figure 12 - Présentation du code Python générant le tableau servant à créer l'histogramme des tirages

En ajout à cette section, il y a également une autre fonction appelée 'reset_histo_count()' permettant de restaurer entièrement toutes les valeurs du tableau histo_count[] à 0.

Celle-ci est utilisée notamment lors de plusieurs tirages pendant une seule section afin d'éviter que le tableau ne garde toute les valeurs des précédents tirages et donc rendre un histogramme faussé..

CODE DE GÉNÉRATION DE L'HISTOGRAMME EN SOIT:

La librairie Matplotlib est utilisée pour la génération simplifiée de graph en python.

On passe en argument de cette fonction le tableau histo_count[] présenté plus tôt ainsi qu'un second paramètre provenant d'un input plus tôt dans le script par l'utilisateur.

Ce second paramètre permet à l'utilisateur de choisir si oui ou non il veut générer un graph avec le tirage effectué ou les données chargées mais également s' il souhaite avoir ce graph simplement affiché à l'écran ou bien enregistré en tant que PNG dans le dossier de travail du programme.

Il est à noter que la capture ci dessous (cf. Figure - 13), ne présente que la fonction en soit, un tableau est préalablement créé avant explication nommé NumberArray[] et contenant à l'intérieur 45 valeurs allant de 1 à 45. Ce tableau est utilisé pour générer l'axe des abscisses de l'histogramme. Il aurait été possible de le créer à l'aide d'une boucle (ce qui n'est pas le cas dans mon code, voir ligne 4 de MATLIB_HISTO.py) mais cela ne change pas grand chose.

```
def Show_Graph(data, comportement):
    plt.hist(NumberArray, bins=45, weights=data, color='orange', edgecolor='black')#Options esthetiques du graph
    # © REMERCIEMENT A THIBAULT COSENZA [RT12] POUR L'ARGUMENT bins=45 PERMETTANT UN BLOCAGE MAX DES COLONNES
    plt.xlabel('Nombre d\'itérations des valeurs')#label Abscisses
    plt.ylabel('Valeur')#label Ordonnées
    plt.title('Histogramme des numéros tirés')#titre global

    if comportement == "GRAPH":#SAUVEGARDE GRAPH EN PNG
        name_file = str(input("\nMerci d'entrer le nom du fichier pour le png : \n"))#Selection nom du png a sauvegarder
        plt.savefig(name_file)
    elif comportement == "SHOW":#AFFICHE GRAPH EN POP-UP
        print("\nAffichage du graph...\n")
        plt.show()#affiche le graph dans un nouvelle fenetre
```

Figure 13 - Présentation du Code Python générant un histogramme à partir de données passées en paramètre

Remerciement spécial à Thibaut COSENZA pour les explications en détail de la fonction .hist() de MatplotLib. Après de nombreuses recherches sur leur documentation, je n'arrivais pas à obtenir exactement le résultat que je souhaitais, en l'occurrence; des colonnes fixes toujours de même taille.

OBSERVATIONS D'UN HISTOGRAMME ISSU D'UN NOMBRE ÉLEVÉ DE TIRAGES:

Après avoir effectué un tirage de 50000 fois 5 nombres, j'ai sélectionné l'option me permettant de générer un histogramme (cf. Figure 14).

J'ai également décidé d'imprimer le résultat dans la console des données brut du tableau histo_count[] (cf. Figure 15).

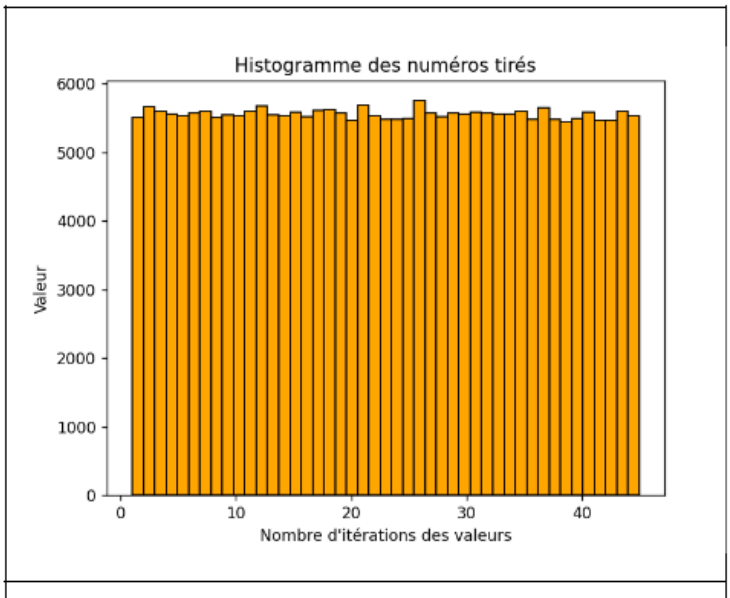


Figure 14 - Histogramme généré par 50 000 tirages de nombres aléatoirement

Il est très clair lorsque l'on regarde l'histogramme ci-dessus qu'à partir d'un certain nombre de tirage, chaque valeur sort à peu près autant de fois qu'une autre.

Ce résultat est certes déjà très parlant avec les données brut d'un tirage (cf. Figure 15) mais il l'est d'autant plus grâce à l'utilisation d'un histogramme pour visualiser les résultats.

En effet, si l'on venait à tracer une droite faisant la moyenne de de la hauteur de chaque colonne, alors celle ci est quasiment parallèle à l'axe des abscisses, ceci prouve donc bien que le résultat est bel et bien aléatoire et que chaque numéro tiré à le même nombre de chance de sortir qu'un autre !

OBSERVATION DES DONNÉES BRUTS ET COMPARAISON AVEC L'HISTOGRAMME:

```
Que souhaitez vous faire ?

 A - Lancer un tirage
 B - Changer mes paramètres de tirage
 C - Charger un fichier de données
 D - Afficher les paramètres actuels
 E - Afficher l'histogramme du dernier tirage

e
Nombre de fois 1 : 5586
Nombre de fois 2 : 5663
Nombre de fois 3 : 5595
Nombre de fois 4 : 5563
Nombre de fois 5 : 5529
Nombre de fois 6 : 5575
Nombre de fois 7 : 5599
Nombre de fois 8 : 5500
Nombre de fois 9 : 5543
Nombre de fois 10 : 5537
Nombre de fois 11 : 5595
Nombre de fois 12 : 5676
Nombre de fois 13 : 5545
Nombre de fois 14 : 5531
Nombre de fois 15 : 5583
Nombre de fois 16 : 5516
Nombre de fois 17 : 5608
Nombre de fois 18 : 5629
Nombre de fois 19 : 5576
Nombre de fois 20 : 5470
Nombre de fois 21 : 5685
Nombre de fois 22 : 5526
Nombre de fois 23 : 5473
Nombre de fois 24 : 5474
Nombre de fois 25 : 5402
Nombre de fois 26 : 5754
Nombre de fois 27 : 5569
Nombre de fois 28 : 5522
Nombre de fois 29 : 5566
Nombre de fois 30 : 5557
Nombre de fois 31 : 5582
Nombre de fois 32 : 5571
Nombre de fois 33 : 5552
Nombre de fois 34 : 5558
Nombre de fois 35 : 5600
Nombre de fois 36 : 5473
Nombre de fois 37 : 5649
Nombre de fois 38 : 5480
Nombre de fois 39 : 5438
Nombre de fois 40 : 5498
Nombre de fois 41 : 5580
Nombre de fois 42 : 5469
Nombre de fois 43 : 5462
Nombre de fois 44 : 5597
Que souhaitez vous en faire ?

     A - Générer un graph
     B - Réinitialiser l'histogramme
     C - Ne rien faire (retour menu)
```

Figure 15 - Présentation dans la console Python des résultats du nombre de fois ou chaque numéro à été tiré

Comme nous pouvions en attester sur l'histogramme, on voit également très clairement sans celui-ci que chaque valeur possible à été tirée environ 5500 x chacune soit environ 250 000 nombres tirés en tout (5 nombres x 50000 tirages).

Je profite également de ce segment pour présenter plus en détail l'interface de cette section du programme.

Une fois le tirage effectué, nous avons plusieurs options disponibles pour en relancer un nouveau, modifier les options du tirage actuel, voir les paramètres actuels, charger un nouveau fichier et notamment "Afficher l'histogramme du dernier tirage", qui à été sélectionné dans l'exemple ci-contre.

On peut ensuite soit réinitialiser le tableau de l'histogramme actuel, s'en servir pour en générer un nouveau graph (utile si l'option n'était pas active jusqu'alors) et simplement retourner au menu principal.

Cette partie du projet est clairement celle qui m'aura le plus passionné étant donné que cela m'a permis de gérer de grandes quantités de données mais également de faire en sorte que tout soit fait de manière automatisée grâce aux inputs de l'utilisateur.

f. Chargement de données

Quelle est la différence entre un format binaire et un format lisible humainement ?

Un format binaire est un format de fichier qui repose sur l'utilisation de bits (soit 0 soit 1). La grande force de ce format est donc qu'il est fait pour être lu par des machines et celles-ci pourront donc le lire plus rapidement qu'un autre format.

C'est d'ailleurs pour cela que l'on utilise ce format notamment pour les fichiers vidéos, audio et les images. Ces données ont besoin d'être lues rapidement et quoi de mieux que le binaire pour effectuer cette tâche ?

Le problème est que ce type de format ne sera forcément pas lisible par un humain, en effet, d'après mes recherches (cf. Sources), même via l'utilisation de logiciels spécialisés il est très compliqué de manipuler ce type de données.

C'est là qu'entre en jeu les formats lisibles plus facilement par des humains. En transformant les bits en caractères de type lettres / chiffres et symboles, nous pouvons facilement lire et manipuler ce genre de fichiers. L'ASCII en est un bon exemple.

Les formats lisible humainement sont donc parfait pour nous pour rédiger des documents, des feuilles de calculs, des sites web, en bref, n'importe quoi ou des fichiers de données ont besoin d'être rédiger, lu et modifier facilement.

N.B : les fichiers lisibles humainement permettent notamment l'indentation ce qui rend la lecture simple à l'œil humain. Ceci importe cependant très peu à une machine.

Quels seraient alors deux formats lisibles humainement en guise d'exemple ?

JSON

Le Json (= Javascript Object Notation) est un format standard d'organisation de données créé en l'an 2000.

Celui-ci utilise des caractères lisibles humainement pour sauvegarder des données dans un forme d'arborescence.

Il a été initialement créé pour le langage de programmation Javascript mais sa relative simplicité d'utilisation et son efficacité ont fait qu'il est désormais utilisable par quasiment tous les langages de programmation grand-public disponibles actuellement.

Voici ci-dessous un exemple basique de l'organisation de données au sein d'un fichier en .json :

```
{
    "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 DocBook.",
                                                "GlossSeeAlso": ["GML", "XML"]
                    },
                                        "GlossSee": "markup"
                }
            }
        }
    }
}
```

Figure 16 - Exemple de fichier Json et son système d'indentation

Comme visible ci-dessus, toute la force du Json repose sur sa lisibilité simple grâce à une forme d'arborescence. Chaque valeur peut également être nommée et ces valeurs peuvent être n'importe quel type de variable (int, str, array…).

C'est pourquoi le Json est par ailleurs désormais très utilisé dans les bases de données en tant réel car son arborescence rend la navigation au sein d'un fichier très simple et surtout rapide (si celui-ci est bien organisé bien sûr).

CSV

Le CSV (= Comma-Separated Values) est un type de format d'organisation de données standardisé créé en l'an 1972. Ceci en fait l'un des plus vieux format de données existant.

Le CSV consiste en un fichier rempli de texte avec chaque valeur insérée étant séparée par une virgule.

Ce qui est intéressant avec le format CSV est qu'il s'agit pour beaucoup de monde plus d'une convention que d'un réel format. En effet on pourrait très bien utiliser un fichier texte de base et choisir de remplacer les virgules par des points-virgules ou n'importe quel autre caractère en guise de séparateur.

Le format .csv existe bien mais ce n'est pas obligatoire pour faire fonctionner celui-ci.

La simplicité de ce type de format en fait un format fort apprécié et très facile d'utilisation mais c'est aussi là que se situent ses limites.

Parcourir un fichier extrêmement volumineux de type CSV implique de passer et de vérifier chaque valeur une par une là où d'autres formats avec des données imbriquées dans d'autres permet de simplement 'passer' rapidement les endroits où l'on sait que l'information ne s'y trouvera pas.

Voici ci-dessous un exemple basique de fichier CSV généré par ailleurs par le programme de tirage loto.

N.B : l'ajout de saut de ligne est utilisé dans cette capture pour une lisibilité accrue mais n'est pas nécessairement un standard du format !

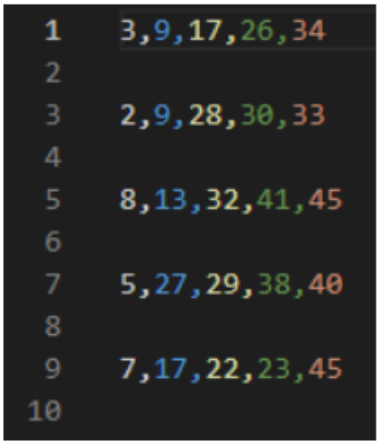


Figure 17 - Exemple de fichier ordonné selon le format CSV

Cette section présente maintenant chacune des fonctions utilisées pour charger des données en format Json, text, csv et binaire.

J'ai utilisé dans mon programme une fonction par type de fichier afin de s'assurer du bon chargement de celui-ci. Chaque fonction fonctionne selon la même organisation :

- Le nom de la fonction est "Charge_" + le type de format
- Chaque fonction prend un seul argument : le nom du fichier à charger
- Chaque fonction renvoie au final l'entièreté du contenu du fichier sous une forme de tableau contenant à chaque index du tableau le tirage correspondant.
- Utilisation de la fonction open() + argument 'r' afin de spécifier qu'on souhaite lire le fichier

CODE PYTHON DU CHARGEMENT DE DONNÉES SOUS FORMAT CSV :

```python
def Charge_Csv(nom_fichier):
    data = []
    with open(nom_fichier, 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            if row != []:
                for i in range(len(row)):
                    row[i] = int(row[i])
                data.append(row)
    return data
```

Figure 18 - Fonction Python utilisée pour charger un fichier CSV

Il est à noter que cette fonction à un ajout spécifique au CSV, lors de l'utilisation de la fonction 'csv.reader', celle ci ne lit qu'une seule ligne du fichier, d'où l'ajout d'une boucle de la longueur du nombre de rangée du fichier afin de charger l'entièreté du fichier et enregistrer ca dans le tableau final à retourner.

CODE PYTHON DU CHARGEMENT DE DONNÉES SOUS FORMAT JSON :

```python
def Charge_Json(nom_fichier): # Param
    with open(nom_fichier, 'r') as f:
        data = json.load(f)
    return data
```

Figure 19 - Fonction Python utilisée pour charger un fichier JSON

Rien à dire de spécial ici si ce n'est l'utilisation de la fonction json.load() grâce à la librairie Json.

Il n'y a ici pas besoin de le faire ligne par ligne car cette fonction charge directement toute l'indentation du fichier que l'on ouvre.

CODE PYTHON DU CHARGEMENT DE DONNÉES SOUS FORMAT TXT :

```python
def Charge_Txt(nom_fichier): # Param = nom du fichier a charger
    data = []
    with open(nom_fichier, 'r') as f:
        for line in f:
            #ligne ci dessous permet de transformer les données et les utiliser ligne par
            data.append([int(x) for x in line.replace('[','').replace(']','').split()])
    return data
```

Figure 20 - Fonction Python utilisée pour charger un fichier TEXT

Il a été nécessaire pour cette fonction de procéder comme avec le CSV, c'est-à -dire charger tout le document ligne par ligne jusqu'à la fin et également supprimer certains caractères sans quoi ceux-ci seraient présents dans le tableau de données final.

Cela ne représente pas un problème en règle générale mais ici il est nécessaire de s'en débarrasser ou bien il ne sera pas possible de manipuler les données avec mes fonctions suivantes.

CODE PYTHON DU CHARGEMENT DE DONNÉES SOUS FORMAT BINAIRE :

```python
def Charge_Binary(nom_fichier): # Param = nom du fichier a charger
    data = []
    with open(nom_fichier, 'rb') as f:
        try:
            while True:
                data.append(pickle.load(f).tolist())
        except EOFError:
            pass
    return data
```

Figure 21 - Fonction Python utilisée pour charger un fichier BINAIRE

J'utilise pour cette tâche la librairie 'Pickle' disponible sur Python, cette librairie permet une gestion simplifiée des formats binaires qu'il s'agisse de les charger ou de les enregistrer.

J'utilise ici un nouveau moyen découvert au moment de la rédaction de cette partie du code, le 'Try / Escept' qui permet de prévenir l'arrivée d'une erreur et de s'y préparer en conséquence.

En l'occurrence je parcours le fichier binaire en boucle jusqu'à ce qu'une erreur soit déclarée une fois la fin du fichier atteinte.

L'avantage de pickle est que les données une fois sorties du fichier binaire sont automatiquement transformées en caractères lisibles par un humain et donc utilisables dans la suite du programme.

N.B : Pour le binaire on utilise open() avec l'argument 'rb' au lieu de 'r' pour 'read binary'.

g. Sauvegarde de données

Comme pour le chargement des données, j'ai utilisé dans mon programme une fonction par type de fichier à enregistrer. Ces fonctions respectent globalement toutes les mêmes grandes lignes, soit :

- Le nom de la fonction est "Sauv_" + le format
- Chaque fonction prend deux paramètres, le premier est le tableau de données à insérer et le second est le nom du fichier sous lequel on souhaite l'enregistrer.
- Si le fichier à charger existe déjà pour une raison x ou y, alors le nom sera automatiquement changé pour garder celui d'origine intact.
- Chaque fonction fait en sorte d'enregistrer toutes les données en respectant les conventions du format en question.
- Utilisation de la fonction open() avec l'argument 'a' pour préciser qu'on souhaite ajouter au fichier et non pas écraser les données d'ores et déjà présentes.

CODE PYTHON DE SAUVEGARDE DE DONNÉES SOUS FORMAT CSV:

```python
def Sauv_Csv(data, nom_fichier):
    if os.path.isfile(nom_fichier) == True:
        nom_fichier = nom_fichier.replace('.csv', '_bis.csv')
    with open(nom_fichier, 'a') as file:
        writer = csv.writer(file)
        writer.writerow(data)
```

Figure 22 - Fonction Python de sauvegarde de données sous format CSV

Pour cette fonction j'utilise toujours la librairie csv de Python pour la manipulation simple de fichier CSV.

Il est à noter que dans le programme final, la section permettant de vérifier l'existence du fichier existant et remplaçant le nom du fichier à enregistrer à été retirée, en effet pour une raison inconnue ceci causait des problèmes dans le code mais par manque de temps je n'ai pas pu l'implémenter de nouveau.

Ceci m'embête beaucoup mais j'ai au moins pu comprendre la théorie derrière le fonctionnement de ce genre d'options et j'espère donc pouvoir l'utiliser dans un projet prochain.

CODE PYTHON DE SAUVEGARDE DE DONNÉES SOUS FORMAT JSON:

```python
data_Json_Formatted = [] #Cas spécial pour le json, sauvegarde d


def Sauv_Json(data, nom_fichier):
    if os.path.isfile(nom_fichier) == True:
        nom_fichier = nom_fichier.replace('.json', '_bis.json')
    with open(nom_fichier, 'w') as f:
        json.dump(data, f)
```

Figure 23 - Fonction Python de sauvegarde de données sous format JSON

La sauvegarde en Json est un cas un peu spécial car il est compliqué d'enregistrer chaque tirage l'un après l'autre dans le fichier, ceci cause des problèmes dans la syntaxe final du fichier.

C'est pourquoi j'ai à la place décidé de créer un tableau à part data_Json_Formatted[] dans lequel on enregistre chaque nouveau tirage qui arrive avant de tout insérer directement dans le fichier Json.

C'est également pour ca qu'on utilise avec la fonction open() l'argument 'w' qui écrase le tout puisqu'il ne sera pas possible dans mon cas d'ajouter des données Json à un fichier pré-existant !

CODE PYTHON DE SAUVEGARDE DE DONNÉES SOUS FORMAT TEXT:

```python
def Sauv_Txt(data, nom_fichier):
    if os.path.isfile(nom_fichier) == True:
        nom_fichier = nom_fichier.replace('.txt', '_bis.txt')
    Saving_Data = open(nom_fichier, 'a')  # use 'a' to not overwrite anything
    Saving_Data.writelines(str(data) + '\n')
    Saving_Data.close()
```

Figure 24 - Fonction Python de sauvegarde de données sous format TEXT

Pour l'enregistrement sous format text, on enregistre directement après chaque tirage la ligne tirée que l'on intègre par dessus les données déjà présentes dans le fichier.

On voit ceci facilement avec l'argument 'a' de la fonction open().

CODE PYTHON DE SAUVEGARDE DE DONNÉES SOUS FORMAT BINAIRE:

```python
def Sauv_Binaire(data, nom_fichier):
    if os.path.isfile(nom_fichier) == True:
        nom_fichier = nom_fichier.replace('.bin', '_bis.bin')
    with open(nom_fichier, 'ab') as f:
        pickle.dump(data, f)
```

Figure 25 - Fonction Python de sauvegarde de données sous format BINAIRE

Comme pour le chargement de données binaires, je ne peux faire sans l'utilisation de la librairie Pickle qui permet de transformer les données humainement lisibles en format binaire avant d'envoyer le tout vers le fichier portant le nom fourni par l'utilisateur.

L'autre léger changement par rapport aux formats plus classiques est l'argument de open qui est ici 'ab' pour 'add binary' au lieu de simplement 'a'.

h. Sélection de filtres pour le tirage

Je ne passerais que rapidement sur cette section, mais j'ai prévu pour mon programme une fonction qui permet de modifier les préférences du tirage par l'utilisateur, celle-ci est déclenchée automatiquement dès le premier tirage mais peut également être appelée de nouveau à chaque fois que l'utilisateur souhaite modifier ses paramètres de tirage actuel.

Cette fonction permet de gérer notamment les options suivantes :

- Utilisation d'une graine (oui ou non)
    - Possibilité d'entrer une graine spécifique si oui
- Type de fichier à utiliser pour l'enregistrement des données (avec ou sans)
    - CSV
    - TXT
    - BIN
    - JSON
        - Entrer le nom du fichier à utiliser
- Type de triage pour les tirages (avec ou sans)
    - Tri fusion
    - Tri insertion
    - Tri cocktail
- Générer un Histogramme des valeurs tirées (avec ou sans)
    - Sauvegarde en png
    - Affichage en pop-up

Une capture de cette partie du code est insensée au vu de sa taille, cette fonction est cependant disponible sur le fichier 'main.py' de la ligne 247 à 324 du script.

Il est également à noter qu'une seconde fonction vient compléter celle-ci s'appelant 'recap_Filtres()' et qui affiche simplement les paramètres enregistrés actuellement dans le programme (ligne 204 à 244).

i. Tirage du loto

Durant l'exécution du programme, une fois les paramètres de tirage choisis et le tirage prêt à être fait, la fonction plusieursTirages() (cf. Figure 26) va prendre en argument le nombre de tirage a effectué.

A chaque tour de la boucle de cette fonction, celle-ci va alors appeler la fonction tirage_des_nombres() (cf. Figure 27) qui va se charger de tirer les 5 nombres via l'utilisation de numpy et en se basant sur tous les paramètres du tirage fourni par l'utilisateur.

Cette fonction imprime également chaque tirage dans la console mais ceci est assez anecdotique étant donné que les données seront lisibles par l'utilisateur à n'importe quel autre moment et sous d'autres formes s'il en décide ainsi.

C'est également la fonction tirage_des_nombres() qui va se charger d'appeler la fonction add_Count(), mentionnée plus tôt dans ce rapport, qui enrichit le tableau de l'histogramme.

Ces deux fonctions sont initiées par une troisième fonction StartUp() (cf. Figure 28) qui est la dernière étape avant le tirage et qui permet à l'utilisateur de spécifier combien de tirages il souhaite effectuer.

L'organisation globale de tout ceci peut sembler un peu anarchique mais le fait de tout segmenter ainsi en plein de fonctions différentes me permet de modifier mon script continuellement sans que cela n'impacte l'entièreté du script.

Si tout était dans une seule grande fonction, il n'y aurait alors pas de possibilité de récursivité dans le code ni d'ajouter des options sans devoir absolument tout repenser dans le programme.

NB : J'ai également ajouté une vérification du type de valeur donnée à StartUp() afin de s'assurer que le nombre de tirage à effectuer est bien un nombre valide.

CODE PYTHON DU TIRAGE :

```python
def StartUp():
    StartSamples = int(input("Combien de séquences souhaitez vous générer ?\n"))
    print(type(StartSamples))
    if type(StartSamples) is int:
        plusieursTirages(StartSamples)
    else:
        print(f"{StartSamples} n'est pas une valeure correcte")
        StartUp()
```

Figure 28 - Fonction StartUp() à la base du lancement du tirage

```python
def plusieursTirages(Nbr):
    global data_Json_Formatted
    global JsonName
    global Graph_Option
    for x in range(Nbr):
        tirage_des_nombres(x)

    if saved_format == 'C':
        Sauv_Json(data_Json_Formatted, JsonName)

    if Graph_Option == "GRAPH" or Graph_Option == "SHOW":
            Show_Graph(histo_count, Graph_Option)
```

Figure 26 - Fonction plusieursTirages(Nbr) en Python

Cette fonction prend en argument le nombre de tirage à effectuer, cette valeur lui est transmise via la fonction StartUp()

```
def tirage_des_nombres(loop):
    global filter_selection
    global saved_format
    global JsonName
    global CsvName
    global TxtName
    global BinName
    global data_Json_Formatted

    data = np.random.choice(range(1, 46),5, replace=False)
    add_Count(list(data))
```

Figure 27 - Fonction tirage_des_nombres()

L'entièreté de la fonction n'est pas présente dans cette capture car ce n'est ici pas ce qui nous intéresse.

Il est surtout important ici de se concentrer sur la 9ème ligne de la capture. En effet, c'est qui génère les 5 chiffres de chaque tirage en prenant.

Voyons en détail comment celle-ci est composée :

Section de la ligne scrutée	Effet sur le tirage
np.random.choice	Le mot-clef 'random' spécifie que l'on souhaite une suite de nombre tirée aléatoirement.
(range(1,46)	On indique ici à Numpy que l'on souhaite que les 5 nombres tirés soient compris entre 1 et 45 (46 n'étant pas inclu dedans)
, 5 ,	On indique ici que l'on souhaite un tirage de 5 nombres
replace = False	On indique ici que si un nombre à été tiré, il ne pourra pas être tiré de nouveau.

Remerciement spécial à Pierre Bourger pour la découverte du 'replace=False' permettant de retirer aux nombres déjà tirés de sortir de nouveau.

4. BIBLIOGRAPHIE

a. Sources utilisées / remerciements

Mes principales sources utilisées pour ce rapport et la création du script auront été les suivantes :

- Khan Academy [https://www.khanacademy.org/]
- CodeCademy [https://www.codecademy.com/]
- Pierre GIRAUD [https://www.pierre-giraud.com/]
- MDN WebDocs [https://developer.mozilla.org/en-US/]
- W3S [https://www.w3schools.com/]
- 'Analyse de données', par Guillaume Broc.
- 'Le guide du débutant Raspberry Pi', par Gareth Halfacree
- Mr Bindel Sebastien, "Aide à la rédaction d'un rapport"
- Pierre Bourger
- Thibaut Cosenza

b. Aide reçue & recherches effectuées

L'entièreté du code et de ce rapport a été réalisée par mes soins, l'aide reçue est notée en commentaire aux endroits concernés dans le code et visible ci-dessus pour les ressources et personnes m'ayant aidé pour la création du programme.

J'ai utilisé pour m'aider beaucoup de ressources de sites d'aide à l'apprentissage de la programmation, notamment Khan Academy, CodeCademy, PIERRE GIRAUD (site web), W3S, et MDN Mozilla documentations.

L'utilisation de chatGPT à été entièrement proscrite pour la rédaction de ce programme et du rapport. Remerciements spéciaux à Pierre BOURGER, Thibaut COSENZA, Mr Ismail BENNIS et BINDEL Sébastien pour leur aide et ressources / cours.

La plupart des recherches effectuées étant surtout par rapport à la syntaxe du python et son fonctionnement spécifique à certaines situations, ayant déjà un peu d'expérience dans le Web-Design, les concepts de base de la programmation n'étaient pas une découverte mais c'était le cas pour tout ce qui touche au python en soi.

5. CONCLUSION

a. Compétences acquises

- Utilisation d'IDE spécifiques au python tel que PyCharm & Spyder
- Utilisation du Python pour des programmes simples
- Utilisation des librairies Numpy & Matplotlib
- Utilisation de pseudo-code pour mettre en forme un algorithme avant de l'implémenter dans le langage souhaité
- Utilisation des mathématiques pour calculer des probabilités
- Gestions de grandes quantitées de données numériques
- Utilisation de librairies Python
- Sauvegarder des données sous différents formats
- Charger des données dans plusieurs formats différents
- Compréhension de la notion de niveau de complexité d'un algorithme
- Systématiquement citer ses sources et nommer les illustrations utilisées
- Gestion de son temps sur un projet de moyenne durée

b. Difficultées rencontrés

- Utilisation du Json dans un contexte autre que via le Javascript
- Utiliser un langage de programmation sans signe clair de délimitation de fonctions autre que l'indentation
- Rédiger un rapport de manière claire et concise au possible malgré sa taille
- Sauvegarder des données d'une manière standardisée afin de pouvoir les utiliser partout par la suite

c. Synthèse du projet & ressenti

Je ne pense pas avoir déjà été aussi impliqué dans un projet scolaire de ce genre, le fait d'avoir eu un objectif précis à réaliser m'a permis de découvrir de multiples aspects du Python et des algorithmes en général que je n'aurais clairement pas vu de cette manière sans cette SAE.

Certaines parties auront été plus difficiles à mettre en œuvre que d'autres mais cela m'a également forcé à développer un peu de résilience lorsque le code ne marche pas comme on le souhaite.

J'aime également à penser que le fait de m'être forcé à faire en sorte que le programme soit une grande boucle infinie m'aura également appris à penser à l'avance à l'organisation du programme au lieu de simplement commencer et d'ensuite gérer les problèmes au fur et à mesure qu'ils se présentent.

Étant au départ assez réticent à l'idée d'utiliser le langage Python que je ne connaissais jusqu'alors pas du tout, ce projet m'aura finalement permis de découvrir ce qui fait la force de ce langage mais également ses limitations.

En effet le fait qu'il n'y ai pas à préciser explicitement à chaque fois le comportement attendu d'une variable ou d'une fonction semblablement au Java ou au langage C m'a mis plusieurs fois des coups d'arrêt ne comprenant pas pourquoi telle ou telle variable changeait de type sans que je ne le lui ai demandé.

En bref, je suis très satisfait de la tournure de ce projet et des compétences qu'il m'aura permis d'acquérir, je ne suis également pas peu fier du résultat final du programme et du rapport et je pense très honnêtement continuer de travailler dessus sur mon temps libre afin de le perfectionner et d'y associer une véritable interface utilisateur via Django et la sauvegarde des données sur des bases de données en ligne.

En vous remerciant pour l'attention que vous aurez porté à ce rapport.

6. ANNEXES

a. Solutions à l'exercice No 3

Question : Quinze chevaux participent à une course. Si cette course est un tiercé combien y a t'il de paris possibles ? Un dimanche matin, un parieur prend 5000 paris différents pour le tiercé de l'après-midi. Que peut-on dire du nombre de chevaux engagés dans la course ?

En sachant qu'il y a 15 chevaux dans la course et qu'un tiercé inclut 3 chevaux sur lesquels parier :

Je peux utiliser la formule de combinaison ce qui me donne : $\frac{15!}{(3!(15-3)!)} = 455$

Je sais désormais qu'il y a donc 455 paris possibles pour 15 chevaux participants à la course.

Je ne suis cependant pas capable en l'état de déterminer combien de chevaux sont engagés dans la course du dimanche matin.

b. Solutions à l'exercice No 4

Question : Un joueur lance cinq pièces de monnaie. Quelle est la probabilité (a) d'avoir uniquement des faces, (b) d'avoir exactement trois faces, (c) d'avoir au moins trois faces ? On considère ici PFFFP et FFFPP comme des résultats distincts.

La probabilité de ne tomber que sur des faces est de : $(\frac{1}{2})^5 = \frac{1}{32}$

La probabilité de tomber sur exactement trois faces est de :

$(5!/(3! * 2!)) * (1/2)^3 * (1/2)^2 = 10 * (1/2)^3 * (1/2)^2 = 10 * (1/32) = 1/3$

c. Probabilités relatives au loto

Question : Donner le nombre de combinaisons possibles pour ce jeu du loto quand l'ordre est pris en compte.

En utilisant la formule des combinaisons, on trouve 1 221 759 arrangements possibles mais uniquement lorsque l'ordre n'est pas pris en compte.

Question : Donner le nombre de combinaisons possibles pour ce jeu du loto quand l'ordre n'est pas pris en compte.

En utilisant la formule des arrangements, on trouve 146 611 080 arrangements possibles mais uniquement lorsque l'ordre est pris en compte.

Question : Donner la probabilité d'avoir les 5 bon numéros (quelque soit l'ordre).
N/A

Question : Donner la probabilité d'avoir les 5 bon numéros (en respectant l'ordre du tirage).
N/A