Quentin SEURON - Tom CARLES - Clément LE ROY
Master Expertise Économique

Projet Python :
Le sport au sein du cadre scolaire

Nous avons hérité du projet d'Emma qui traite du sport au sein du cadre scolaire. Notre travail consiste à analyser dans la première partie, les réponses d'étudiants d'une même classe puis, dans un deuxième temps, de répondre à un sondage lancé par les professeurs de l'établissement scolaire et, dans un dernier temps, de simuler un match de basket.

Contexte :
D'une façon générale, en classe de 1ère et Terminale, les élèves doivent choisir un programme sportif composé de trois sports prédéfinis. Une fois le choix effectué, ils suivront un enseignement sportif basé sur leur sélection. Un menu se déroule le temps d'un trimestre.

Préambule :
La base de données fournie nous renseigne sur le numéro de l'étudiant, son sexe, le menu choisi pour l'année scolaire, son sport favori tous menus confondus, son sport le moins aimé tous menus confondus aini que la satisfaction ou pas de l'étudiant sur le choix des menus.

Explications importantes :
Pour la majorité des questions de la partie 1, nous allons utiliser la lecture du document "DonnéesClassePrincipale.txt". Cette lecture de fichier nous donnera une liste ([]) dans laquelle chaque élément correspond à une ligne du fichier txt, c'est-à-dire, chaque étudiant. Nous avons appelé cette liste "s". On peut expliquer cela de manière simplifiée comme suit : [étudiant1, étudiant2, …, étudiant33].
Afin de pouvoir avoir accès aux réponses de tous les étudiants plus facilement, nous allons subdiviser cette liste en plusieurs sous listes contenues dans la liste principale. Pour le faire, nous avons coupé la liste "s" à chaque délimitation caractérisée par un espace (" "). Nous avons appelé cette liste "l". Cela donne de manière simplifiée : [[étudiant1, a, b, c], [étudiant2, a, b, c], …. , [étudiant33, a, b, c]] où a,b et c sont les données des étudiants comme expliqué en préambule.
Cette nouvelle liste "l" nous permet d'accéder à l'élément en position l[i][j] où i = {1; … ; 33} et j = {1; … ; 6} où i représente les différents étudiants et j les différentes réponses de chaque étudiant. Finalement, on peut voir cette nouvelle liste "l" comme une matrice ou i représente les lignes et j les colonnes.

```
data = open('DonnéesClassePrincipale.txt','r')
s = data.readlines()
data.close()

l=[]

for i in range(len(s)):
    s[i]=s[i].strip()
    l.append(s[i].split(" "))

print('\ns = ', s, '\n\nl = ', l)
```

```
untitled0.py', wdir='C:/Users/cleme/bureau/Clement/Cours/Fac/Master/Master 1/Introduction to Python/Travail Projet')

s =  ['1010 F A Volley Natation oui', '1011 F A Volley Course oui', '1012 M C VTT Acrosport oui', '1013 M A Basket Natation
non', '1014 F B Basket Acrosport oui', '1015 M A Basket Natation non', '1016 F A Badminton Natation non', '1017 M B Basket
VTT non', '1018 F A Escalade Course non', '1019 F C Course Natation non', '1020 M B Natation Acrosport oui', '1021 F B
Basket Escalade oui', '1022 F A Basket Natation non', '1023 M A Acrosport Escalade oui', '1024 M A Badminton Course oui',
'1025 M A Basket Natation non', '1026 F A Volley Course oui', '1027 F A Basket Natation non', '1028 M B Basket Escalade
non', '1029 F C Course Acrosport oui', '1030 F C Course Natation non', '1031 M C Escalade Natation oui', '1032 F A Basket
Natation non', '1033 M A Basket Natation non', '1034 F C Escalade Volley oui', '1035 M A Basket Natation non', '1036 M A
Basket Natation non', '1037 F C Escalade Basket oui', '1038 M B Basket VTT oui', '1039 F A Basket Natation non', '1040 F C
VTT Acrosport oui', '1041 M A Basket Natation non', '1042 M C Basket Natation non']

l =  [['1010', 'F', 'A', 'Volley', 'Natation', 'oui'], ['1011', 'F', 'A', 'Volley', 'Course', 'oui'], ['1012', 'M', 'C',
'VTT', 'Acrosport', 'oui'], ['1013', 'M', 'A', 'Basket', 'Natation', 'non'], ['1014', 'F', 'B', 'Basket', 'Acrosport',
'oui'], ['1015', 'M', 'A', 'Basket', 'Natation', 'non'], ['1016', 'F', 'A', 'Badminton', 'Natation', 'non'], ['1017', 'M',
'B', 'Basket', 'VTT', 'non'], ['1018', 'F', 'A', 'Escalade', 'Course', 'non'], ['1019', 'F', 'C', 'Course', 'Natation',
'non'], ['1020', 'M', 'B', 'Natation', 'Acrosport', 'oui'], ['1021', 'F', 'B', 'Basket', 'Escalade', 'oui'], ['1022', 'F',
'A', 'Basket', 'Natation', 'non'], ['1023', 'M', 'A', 'Acrosport', 'Escalade', 'oui'], ['1024', 'M', 'A', 'Badminton',
'Course', 'oui'], ['1025', 'M', 'A', 'Basket', 'Natation', 'non'], ['1026', 'F', 'A', 'Volley', 'Course', 'oui'], ['1027',
'F', 'A', 'Basket', 'Natation', 'non'], ['1028', 'M', 'B', 'Basket', 'Escalade', 'non'], ['1029', 'F', 'C', 'Course',
'Acrosport', 'oui'], ['1030', 'F', 'C', 'Course', 'Natation', 'non'], ['1031', 'M', 'C', 'Escalade', 'Natation', 'oui'],
['1032', 'F', 'A', 'Basket', 'Natation', 'non'], ['1033', 'M', 'A', 'Basket', 'Natation', 'non'], ['1034', 'F', 'C',
'Escalade', 'Volley', 'oui'], ['1035', 'M', 'A', 'Basket', 'Natation', 'non'], ['1036', 'M', 'A', 'Basket', 'Natation',
'non'], ['1037', 'F', 'C', 'Escalade', 'Basket', 'oui'], ['1038', 'M', 'B', 'Basket', 'VTT', 'oui'], ['1039', 'F', 'A',
'Basket', 'Natation', 'non'], ['1040', 'F', 'C', 'VTT', 'Acrosport', 'oui'], ['1041', 'M', 'A', 'Basket', 'Natation',
'non'], ['1042', 'M', 'C', 'Basket', 'Natation', 'non']]
```

Nous avons, pour chaque question, créé une fonction qui permet de trouver la réponse adaptée. Nous avons écrit une seule fois la liste “l” en début de programme pour nous en servir directement dans les fonctions nécessitant son utilisation. Ainsi, le coût de programmation est moins élevé que si nous l’avions écrit dans chaque fonction.
Pour illustrer nos propos, nous allons tenter d’expliquer cela de manière mathématique :
La subdivision en “liste de liste” peut être approximée en O(n*(2m+1)) avec n = 33, correspondant au nombre d’élèves et m=32 correspondant au nombre de caractères maximal pour un élève (dans le fichier txt). On obtient ainsi une complexité égale à O(33(2*32+1)) ce qui nous donne au maximum 2145 opérations.
On obtient ce nombre d’opérations de la manière suivante :
Pour chaque élément dans s → n exécution

- Appel de la fonction strip sur l’élément s[i] → complexité en la taille de s[i] ici O(m)
- Appel de la fonction split sur l’élément s[i] → complexité en la taille de s[i] ici O(m)
- Appel de la fonction append → complexité en O(1)

Ce qui nous donne pour le corps de la boucle une complexité de O(m+m+1) = O(2m+1) et comme la boucle s’exécute n fois, cela nous donne O(n*(2m+1)).

Si nous avions recopié ces quelques lignes de code dans chaque fonction alors, à chaque fois, nous aurions reproduit les mêmes 2145 opérations. Afin d’économiser de la puissance de calcul et de gagner en vitesse d’exécution, nous avons créé la liste “l” en début de programme pour l’appeler quand nous le souhaitons.

Pour illustrer nos explications, nous avons isolé, lorsque cela n’était pas trop contraignant, certaines questions dans un nouveau fichier Python afin d’avoir une capture d’écran contenant uniquement le code de la question et le résultat obtenu dans la console.

<u>Partie 1</u> : **Classe principale**

<u>Question 1</u> :

Dans cette question, nous devons calculer, grâce à quelques lignes de code, le nombre d’étudiants présents dans cette classe.

```
print('*'*57)

data = open('DonnéesClassePrincipale.txt','r')
s = data.readlines()

l=[]

for i in range(len(s)):
    s[i]=s[i].strip()
    l.append(s[i].split(" "))


# Partie 1

# Question 1

print('La classe contient', len(s), 'étudiants')
print('*'*57)
```

```
In [5]: runfile('C:/Users/cleme/bureau/Clement/Cours/Fac/Master/Master 1/Introduction to Python/
Travail Projet/untitled0.py', wdir='C:/Users/cleme/bureau/Clement/Cours/Fac/Master/Master 1/
Introduction to Python/Travail Projet')
*********************************************************
La classe contient 33 étudiants
*********************************************************

In [6]:
```

Nous avons trouvé que cette classe est composée de 33 étudiants.

Question 2 :

Il nous était demandé dans cette deuxième question de déterminer le nombre de garçons et de filles dans cette classe.

```
print('*'*57)

data = open('DonnéesClassePrincipale.txt','r')
s = data.readlines()

l=[]

for i in range(len(s)):
    s[i]=s[i].strip()
    l.append(s[i].split(" "))


# Partie 1

# Question 2

def gender():
    f=0
    m=0

    for i in range(len(s)):
        if l[i][1] == "M":
            m+=1
        if l[i][1] == "F":
            f+=1

    print("Le nombre de filles est :", f)
    print("Le nombre de garcons est :", m)

gender()
print('*'*57)
```

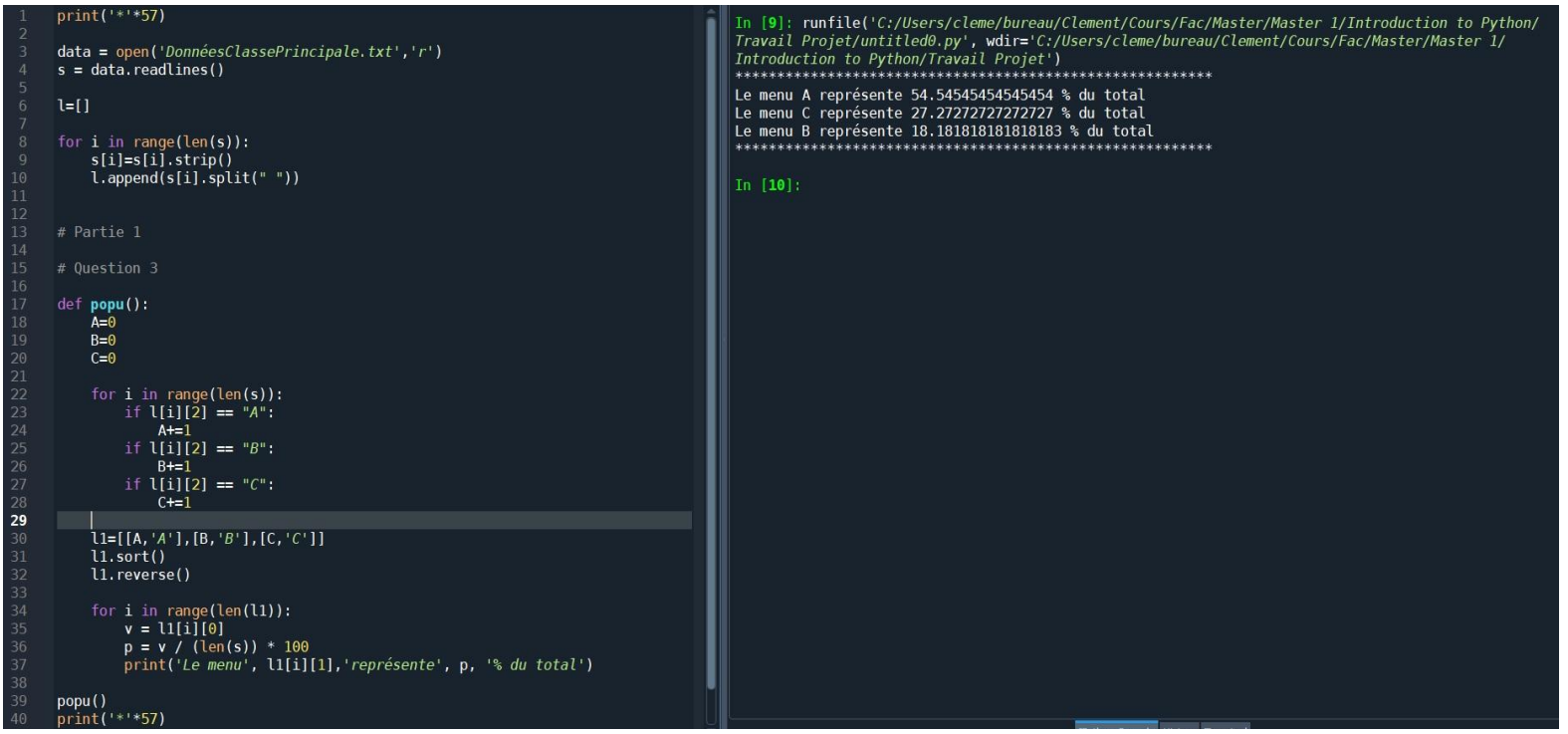
```
In [8]: runfile('C:/Users/cleme/bureau/Clement/Cours/Fac/Master/Master 1/Introduction to Python/
Travail Projet/untitled0.py', wdir='C:/Users/cleme/bureau/Clement/Cours/Fac/Master/Master 1/
Introduction to Python/Travail Projet')
*********************************************************
Le nombre de filles est : 17
Le nombre de garcons est : 16
*********************************************************

In [9]:
```

Nous avons trouvé qu'il y a 17 filles et 16 garçons dans cette classe.

Question 3 :

Afin de classer les menus du plus au moins populaires, nous avons utilisé la liste "l". Dans cette dernière, nous avons compté le nombre de fois que le menu A, menu B et menu C ont été choisis. Les résultats sont stockés dans une nouvelle liste "l1" qui contient trois sous listes (une pour chaque menu). Dans chaque sous liste, il y a le nombre d'occurence d'un menu puis son nom (A, B ou C). Ensuite on trie la liste par ordre croissant. Il est important de préciser que le tri s'effectue sur les premiers éléments des sous listes (ie les nombres, les lettres ne sont pas pris en compte pour le tri). Finalement, il suffit de retourner la liste pour obtenir l'ordre décroissant. A ce moment-là nous pouvons, pour chaque sous liste, calculer le pourcentage correspondant à chaque menu (nombre d'occurrence / 33 *100).



```
print('*'*57)

data = open('DonnéesClassePrincipale.txt','r')
s = data.readlines()

l=[]

for i in range(len(s)):
    s[i]=s[i].strip()
    l.append(s[i].split(" "))


# Partie 1

# Question 3

def popu():
    A=0
    B=0
    C=0

    for i in range(len(s)):
        if l[i][2] == "A":
            A+=1
        if l[i][2] == "B":
            B+=1
        if l[i][2] == "C":
            C+=1

    l1=[[A,'A'],[B,'B'],[C,'C']]
    l1.sort()
    l1.reverse()

    for i in range(len(l1)):
        v = l1[i][0]
        p = v / (len(s)) * 100
        print('Le menu', l1[i][1],'représente', p, '% du total')

popu()
print('*'*57)
```

```
In [9]: runfile('C:/Users/cleme/bureau/Clement/Cours/Fac/Master/Master 1/Introduction to Python/
Travail Projet/untitled0.py', wdir='C:/Users/cleme/bureau/Clement/Cours/Fac/Master/Master 1/
Introduction to Python/Travail Projet')
**********************************************************
Le menu A représente 54.54545454545454 % du total
Le menu C représente 27.27272727272727 % du total
Le menu B représente 18.181818181818183 % du total
**********************************************************

In [10]:
```

Ainsi le menu A représente environ 54% des choix, le menu B environ 27% et le menu C environ 18%.

Question 4 :

Nous devons ici trouver la réponse à deux questions : trouver le sport le plus aimé des étudiants et trouver le sport le moins aimé par ces derniers.
Pour ce faire, à partir de la liste “l”, on va créer une nouvelle liste qui ne contiendra que les données nous intéressant (la colonne 4 pour le sport préféré et la colonne 5 pour le sport le moins aimé). Attention : l’indexation sur Python commence à 0 donc pour Python cela correspond aux indices 3 et 4 pour respectivement le sport plus aimé et le sport le moins aimé. Nous appelons à chaque question cette liste “t”. S’agissant de variables locales, c’est-à-dire inhérentes à la fonction, il n’y a pas de conflit du fait que les variables portent le même nom.
Ensuite, on compte pour tous les éléments de la liste “t” leur nombre d'occurrence que l’on va stocker dans une nouvelle liste “y”. On utilise la même méthode que dans la question 3 où, on a une liste dans laquelle il y a une liste par sport avec son nombre d’occurrence. Il nous reste donc à chercher le sport qui a le plus grand nombre d’occurrences.

On procède exactement de la même manière pour le sport le moins apprécié par les étudiants. Cette donnée est renseignée dans la colonne 5 du fichier fourni pour le projet. Il suffit donc de regarder quel est le sport le plus renseigné pour avoir le sport le moins apprécié par les étudiants.

```python
print('*'*57)

data = open('DonnéesClassePrincipale.txt','r')
s = data.readlines()

l=[]

for i in range(len(s)):
    s[i]=s[i].strip()
    l.append(s[i].split(" "))


# Partie 1

# Question 4

def sport_pref():

    t=[]

    for i in range(len(l)):
        t.append(l[i][3])

    y = [[x,t.count(x)] for x in set(t)]
    y.sort()

    maxi=0
    sport=""

    for i in range(len(y)):
        if y[i][1] > maxi:
            maxi = y[i][1]
            sport= y[i][0]
    print("Le sport le plus apprécié est :" ,sport, "avec", maxi, "votes sur", len(s))

sport_pref()
print('*'*57)
```

```
In [10]: runfile('C:/Users/cleme/bureau/Clement/Cours/Fac/Master/Master 1/Introduction to
Python/Travail Projet/untitled0.py', wdir='C:/Users/cleme/bureau/Clement/Cours/Fac/
Master/Master 1/Introduction to Python/Travail Projet')
**********************************************************
Le sport le plus apprécié est : Basket avec 17 votes sur 33
**********************************************************

In [11]:
```

Le sport préféré des élèves est le basket avec 17 choix sur 33 ce qui correspond à environ 51%.

```python
print('*'*57)

data = open('DonnéesClassePrincipale.txt','r')
s = data.readlines()

l=[]

for i in range(len(s)):
    s[i]=s[i].strip()
    l.append(s[i].split(" "))


# Partie 1

# Question 4

def sport_pas_pref():

    t=[]

    for i in range(len(l)):
        t.append(l[i][4])

    y = [[x,t.count(x)] for x in set(t)]
    y.sort()

    mini=0
    sport=""

    for i in range(len(y)):

        if y[i][1] > mini:
            mini = y[i][1]
            sport= y[i][0]
    print("Le sport le moins apprécié est :" ,sport, "avec", mini, "votes sur", len(s))

sport_pas_pref()
print('*'*57)
```

```
In [11]: runfile('C:/Users/cleme/bureau/Clement/Cours/Fac/Master/Master 1/Introduction
to Python/Travail Projet/untitled0.py', wdir='C:/Users/cleme/bureau/Clement/Cours/Fac/
Master/Master 1/Introduction to Python/Travail Projet')
**********************************************************
Le sport le moins apprécié est : Natation avec 17 votes sur 33
**********************************************************

In [12]:
```

Le sport le moins apprécié par les élèves est la natation avec aussi 17 choix sur 33 donc également environ 51%.

Question 5 :

Pour mesurer le taux de satisfaction des élèves de la classe et répondre à la question, il faut compter le nombre d'élèves ayant répondu “oui”, signifiant qu’ils sont satisfaits des programmes proposés. Il faut ensuite rapporter le nombre de “oui” au nombre total des réponses. Si ce résultat dépasse les deux tiers de la classe (la proportion nous est imposée par le sujet) alors on considère que le taux de satisfaction est acceptable pour les menus de sports proposés.
On utilise la même façon de procéder qu’à la question précédente, c’est-à-dire, extraire une colonne de la liste “l”, ici la colonne 6, et la stocker dans la liste “t”. Il sera beaucoup plus facile de compter le nombre de “oui” dans cette liste “t”.

```
print('*'*57)

data = open('DonnéesClassePrincipale.txt','r')
s = data.readlines()

l=[]

for i in range(len(s)):
    s[i]=s[i].strip()
    l.append(s[i].split(" "))


# Partie 1

# Question 5

def satisfaction():

    t=[]
    c=0

    for i in range(len(l)):
        t.append(l[i][5])

    for i in t:
        if i == "oui":
            c+=1

    print('Le nombre total de \'oui\' obtenu est :', c, '/', len(s))

    if c >= 2/3*len(s):
        print('Le taux de satisfaction est acceptable (', int(c/len(s)*100),'% )')
    else:
        print('Le taux de satisfaction n\'est pas acceptable (', int((len(s)-c)/len(s)*10

satisfaction()
print('*'*57)
```

```
Le nombre total de 'oui' obtenu est : 15 / 33
Le taux de satisfaction n'est pas acceptable ( 54 % )
*********************************************************

In [15]:
```

Il y a 15 élèves qui ont répondu “oui” au sondage ce qui correspond à 54%, ainsi le taux de satisfaction n’est pas acceptable car il est inférieur au taux imposé par le sujet (⅔) environ 66%. Finalement les élèves ne sont globalement pas satisfaits par les menus de sports proposés.

Partie 2 : **Changer les menus**

Contexte :
Après avoir conclu que le Menu B proposé n’était pas acceptable considérant le niveau de satisfaction des élèves, il faut le modifier. Ainsi, un sondage est réalisé afin de déterminer le sport qui remplacerait la natation, sport le plus détesté du menu précédent.

Question 1 :

Le sondage fournit le résultat des votes, le nombre de votes associé à chaque sport constitue les paramètres de la fonction qu'il nous est demandé de créer, dont l'objectif est d'afficher le sport ayant recueilli le plus de votes.
On code de la même manière que le sport le plus aimé. La liste "l" (créée dans cette fonction) est constituée de plusieurs sous listes avec le nom du sport et son nombre de votes. Il faut regarder celui qui a le plus grand nombre de votes et le renvoyer.

```
print('*'*57)

# Partie 2

# Question 1

def choix(saut, hand, escrime):

    l=[['Saut en longueur', saut], ["Handball", hand], ['Escrime', escrime]]
    c=0
    sport=''

    for i in range(len(l)):
        if l[i][1] > c:
            c = l[i][1]
            sport = l[i][0]
    print('Le sport le plus choisi est', sport, 'avec', c, 'votes')

choix(97, 256, 181)
print('*'*57)
```

```
*********************************************************
Le sport le plus choisi est Handball avec 256 votes
*********************************************************

In [16]:
```

Question 2 :
Nous devons maintenant créer un dictionnaire associant chaque menu (A, B et C) aux sports qui le composent.

```
print('*'*57)

# Partie 2

# Question 2

menu = {'Menu A': 'Volley - Accrosport - Badminton', 'Menu B': 'Natation - Basket - Course', 'Menu C': 'VTT - Escalade - Course'}

print(menu.keys())
print(menu.values())
print('\nMenu A :' ,menu['Menu A'])
print('Menu B :' ,menu['Menu B'])
print('Menu C :' ,menu['Menu C'])

print('*'*57)
```

```
*********
dict_keys(['Menu A', 'Menu B', 'Menu C'])
dict_values(['Volley - Accrosport - Badminton',
'Natation - Basket - Course', 'VTT - Escalade -
Course'])

Menu A : Volley - Accrosport - Badminton
Menu B : Natation - Basket - Course
Menu C : VTT - Escalade - Course
*********************************************
*********

In [18]:
```

Nous avons décidé d'utiliser le dictionnaire que nous avons créé afin d'en extraire toutes les clés et toutes leurs valeurs associées. Ensuite, nous avons affiché dans la console de manière plus précise la composition des menus et les sports associés à partir du dictionnaire que nous avions constitué.
On obtient donc de manière plus claire :

- Le menu A est composé du Volley, de l'Acrosport et du Badminton
- Le menu B est composé de la Natation, du Basket et de la Course
- Le menu C est composé du VTT, de l'escalade et de la Course

<u>Partie 3</u> : Simulation d'un match de basket

<u>Contexte</u> :
Nous arrivons à la 3e partie du sujet : la simulation d'un match de basket. Afin de pouvoir faire cette simulation de la manière la plus conforme aux règles du basket, voici un rappel des règles principales :

- Un match oppose deux équipes
- Il y a 3 possibilités pour marquer des points : le panier à 1 point (lancer franc), le panier à 2 points et le panier à 3 points.
- La durée d'un match est de 40 minutes décomposée en 4 quart-temps de 10 minutes.

Nos fonctions pour l'ensemble de cette partie devront être écrites de telle sorte que le score soit mis à jour toutes les minutes du match. Ainsi à chaque minute, les deux équipes peuvent marquer soit 1 point, soit 2 points, soit 3 points mais aussi 0 point si jamais elles ne marquent pas. Après avoir questionné Emma sur la fréquence des paniers, elle nous a indiqué que les deux équipes ne pouvaient marquer qu'un seul panier par minute, peu importe le score.

<u>Question 1</u> :

Nous avons fait en sorte que les points marqués, par chaque équipe, toutes les minutes, soient générés de façon aléatoire, grâce à la fonction "randint" de Python. La probabilité de marquer 0 point est la même que celle de marquer 1 point, 2 points ou 3 points. Nous avons donc une équiprobabilité des résultats possibles. Ainsi les résultats du match ne seront jamais les mêmes si on exécute plusieurs fois la fonction. Pour cela, toutes les minutes, deux nombres sont générés aléatoirement entre 0 et 3 inclus et stockés dans "x" pour l'équipe 1 et dans "y" pour l'équipe 2. Ces nombres nous servent à obtenir le nombre de points qui sont contenus à la position x (ou y) dans la liste "l1". On peut prendre l1[x] et l1[y], grâce aux nombres aléatoirement générés. On obtient aléatoirement un nombre entre 0 et 3 inclus, qui correspond au score marqué par chaque équipe.

Nous répétons cette action tant que le match n'est pas terminé, grâce à une boucle "while". Le match se termine au bout de 40 minutes, il y a donc 40 tours de boucle correspondant aux 40 minutes de jeu.

Une fois le match terminé, on affiche les deux scores finaux que l'on compare pour connaître l'équipe victorieuse. Nous prenons aussi en compte le cas d'égalité entre les deux équipes.

```python
print('*'*57)

# Partie 3

# Question 1

def Partie_Basket():

    from random import randint

    l1 = [0,1,2,3]

    time = 0
    score_equipe1 = 0
    score_equipe2 = 0

    while time != 40:
        x = randint(0,3)
        y = randint(0,3)

        score_equipe1 = score_equipe1 + l1[x]
        score_equipe2 = score_equipe2 + l1[y]

        time = time + 1

    print('Score Final :',score_equipe1, '-',score_equipe2)

    if score_equipe1 > score_equipe2 :
        print('l\'équipe 1 remporte le match')
    elif score_equipe1 == score_equipe2:
            print('Égalité entre les deux équipes')
    else:
        print('L\'équipe 2 remporte le match')


Partie_Basket()
print('*'*57)
```

```
*********************************************************
Score Final : 55 - 49
l'équipe 1 remporte le match
*********************************************************

In [22]:
```

Nous avons décidé d'améliorer un peu cette fonction afin de savoir comment les équipes ont pu arriver à ces scores. Nous avons créé une nouvelle fonction, appelée "Stats_Match" qui reprend exactement le même fonctionnement que la fonction précédente "Partie_Basket" mais, elle compte le nombre de paniers de chaque point marqué par les deux équipes. Nous pouvons donc savoir comment les scores sont composés et avoir des statistiques sur les deux équipes.

```python
# Partie 3

# Question 1

def Stats_Match():

    from random import randint

    l1 = [0,1,2,3]

    time = 0
    score_equipe1 = 0
    score_equipe2 = 0

    pt1 = 0
    pt2 = 0
    pt3 = 0

    pts1 = 0
    pts2 = 0
    pts3 = 0

    while time != 40:
        x = randint(0,3)
        y = randint(0,3)

        score_equipe1 = score_equipe1 + l1[x]
        score_equipe2 = score_equipe2 + l1[y]

        if x == 1:
            pt1+=1
        elif x == 2:
            pt2+=1
        elif x == 3:
            pt3+=1

        if y == 1:
            pts1+=1
        elif y == 2:
            pts2+=1
        elif y == 3:
            pts3+=1

        time = time + 1

    print('Score Final :',score_equipe1, '-',score_equipe2)

    if score_equipe1 > score_equipe2 :
        print('L\'équipe 1 remporte le match')
    elif score_equipe1 == score_equipe2:
            print('Égalité entre les deux équipes')
    else:
        print('L\'équipe 2 remporte le match')

    print('\nStatistiques :\nL\'Équipe 1 à marqué :', pt1, 'paniers à 1 point,', pt2, 'paniers à 2 points,', pt3, 'paniers à 3 points')
    print('L\'Équipe 2 à marqué :', pts1, 'paniers à 1 point,', pts2, 'paniers à 2 points,', pts3, 'paniers à 3 points')

Stats_Match()
```

```
**********************************************************
Score Final : 66 - 67
L'équipe 2 remporte le match

Statistiques :
L'Équipe 1 à marqué : 11 paniers à 1 point, 14 paniers à 2 points, 9 paniers à 3 points
L'Équipe 2 à marqué : 10 paniers à 1 point, 9 paniers à 2 points, 13 paniers à 3 points
**********************************************************

In [23]:
```

IPython Console | History | Terminal

Question 2 :

Dans cette question, nous devons effectuer plusieurs changements sur notre fonction. En effet, nous devons modifier la probabilité de marquer chaque panier de telle sorte qu'elle ne soit plus équiprobable. Nous devons également, comme la consigne le suggère en partie bonus, ajouter du temps additionnel à la fin du match en cas d'égalité.
Nous avons échangé avec Emma, qui nous a apporté des précisions sur l'attendu de la partie bonus. Nous devons continuer le match, en ajoutant 5 min à chaque fois, s'il y a égalité entre les deux équipes jusqu'à ce qu'à la fin des 5 min il n'y ait plus égalité. Ainsi, on pourrait ajouter 5 min de nombreuses fois tant que l'égalité est toujours présente.

Premièrement, la probabilité de marquer un panier : Pour changer la probabilité d'un panier, nous devons respecter les données de l'énoncé. Le 1er tableau ci-dessous donne ces nouvelles proportions et le 2nd tableau, les fréquences cumulées.

Fréquences par équipe \ Paniers	0	1	2	3
Équipe 1	20%	10%	55%	15%
Équipe 2	30%	5%	40%	25%
Fréquences cumulées par équipe \ Paniers	0	1	2	3
Équipe 1	20%	30%	85%	100%
Équipe 2	30%	35%	75%	100%

Ainsi, nous générons un nombre aléatoire chaque minute, grâce à la fonction random. Ce nombre est compris entre 0 et 1 inclus. La liste “l1” contient les résultats du tableau des fréquences ci-dessus pour l’équipe 1 (de même pour la liste “l2”).
Nous allons comparer ce nombre au tableau des fréquences cumulées. Si ce nombre est plus petit que la 1ère colonne de l’équipe 1 (ou l’équipe 2) alors on prendra le score qui correspond à cette colonne. Si ce n’est pas le cas, on regardera si le nombre généré est plus petit que la somme de la 1ère colonne et de la 2e colonne. Si c’est le cas, le score ajouté sera celui dans la 2e colonne. On reproduit exactement le même schéma pour les colonnes 3 et 4.
Prenons un exemple, le nombre généré est e = 0.22681870859658348.
Pour l’équipe 1, ce nombre est plus grand que 0.2, on regarde donc s’il est plus petit que 0.2+0.1=0.3. Comme c’est le cas, l’équipe 1 marque 1 point.
Pour l’équipe 2, le nombre est plus petit que 0.3, l’équipe 2 marque donc 0 point.

En procédant de la sorte, on obtient une répartition différente des probabilités de marquer 0, 1, 2 ou 3 point(s) pour chaque équipe.
On générera ce nombre aléatoire “e” toutes les minutes afin de pouvoir actualiser le score en fonction du panier marqué.

Pour la suite du problème, afin d’alléger notre code, nous avons créé deux fonctions “Match40min” et “prolongations” qui permettent respectivement de générer un match aléatoire de 40 min et de jouer une ou plusieurs prolongations en fonction des besoins.

```
def Match40min():

    global score_equipe1
    global score_equipe2
    global e
    global l1
    global l2
    global l0

    import random

    l0 = [0,1,2,3]
    l1 = [0.2,0.1,0.55,0.15]
    l2 = [0.3,0.05,0.4,0.25]

    score_equipe1=0
    score_equipe2=0
    time=0

    while time != 40:

        e = random.random()

        if e < l1[0]:
            score_equipe1 = score_equipe1 + l0[0]
        elif e < l1[0]+l1[1]:
            score_equipe1 = score_equipe1 + l0[1]
        elif e < l1[0]+l1[1]+l1[2]:
            score_equipe1 = score_equipe1 + l0[2]
        else:
            score_equipe1 = score_equipe1 + l0[3]

        if e < l2[0]:
            score_equipe2 = score_equipe2 + l0[0]
        elif e < l2[0]+l2[1]:
            score_equipe2 = score_equipe2 + l0[1]
        elif e < l2[0]+l2[1]+l2[2]:
            score_equipe2 = score_equipe2 + l0[2]
        else:
            score_equipe2 = score_equipe2 + l0[3]

        time = time + 1

#Match40min()
```

Deuxièmement, le cas d'égalité : il faut rajouter 5 min tant que les scores sont à égalité à la fin du match ou de la prolongation de 5 min.
Pour ce faire, à la fin du temps, on compare les deux scores, tant que l'égalité est présente on lance une boucle de 5 min qui fonctionne exactement comme la partie principale, c'est-à-dire le match de 40 min.

Nous appelons la fonction “prolongations” qui exécute un match de 5 min afin de pouvoir l’appeler sans avoir à recoder la/les prolongation(s) dans d’autres fonctions.

```python
""" On créer maintenant une fonction qui permettra de jouer les prolongations de 5 min """

def prolongations():

    import random
    global prolongation
    global l0
    global l1
    global l2
    global e
    global score_equipe1
    global score_equipe2

    time=0

    while time != 5:

        e = random.random()

        if e < l1[0]:
            score_equipe1 = score_equipe1 + l0[0]
        elif e < l1[0]+l1[1]:
            score_equipe1 = score_equipe1 + l0[1]
        elif e < l1[0]+l1[1]+l1[2]:
            score_equipe1 = score_equipe1 + l0[2]
        else:
            score_equipe1 = score_equipe1 + l0[3]

        if e < l2[0]:
            score_equipe2 = score_equipe2 + l0[0]
        elif e < l2[0]+l2[1]:
            score_equipe2 = score_equipe2 + l0[1]
        elif e < l2[0]+l2[1]+l2[2]:
            score_equipe2 = score_equipe2 + l0[2]
        else:
            score_equipe2 = score_equipe2 + l0[3]

        time = time + 1

#prolongations()
```

Cependant, si à la fin des 40 min il n’y a pas égalité, alors une équipe est déclarée victorieuse. Dans ce cas, il n’y a pas de prolongation. Le score final est obtenu soit à la fin des 40 min, soit à la fin de la/des prolongation(s).

```
Score Final : 71 - 68
l'équipe 1 remporte le match

Il n'y a pas eu de prolongation
```

```
Score : 72 - 72
PROLONGATION
Score : 79 - 79
PROLONGATION
Score : 88 - 88
PROLONGATION
Score : 97 - 97
PROLONGATION
Score Final : 107 - 106
l'équipe 1 remporte le match

Il y a eu : 4 prolongations
```

```
**********
Score : 61 - 61
PROLONGATION
Score Final : 72 - 73
l'équipe 2 remporte le match

Il y a eu : 1 prolongation
**********
```

Notre code permet donc de faire n prolongations (avec n = [0;+infini]), tant que le score des deux équipes n'est pas différent. L'équipe, qui à la fin, a le plus grand score sera l'équipe victorieuse.

Pour aller encore plus loin, nous avons voulu que ce programme fonctionne tant qu'il n'y a pas eu "x" égalité(s). Le nombre "x" serait un nombre choisi par l'utilisateur.
Attention : les résultats sont imprévisibles car générés aléatoirement. Il faut donc être vigilant sur le nombre "x" demandé.
Pour éviter de faire crasher l'ordinateur, nous avons forcé la liste des choix disponibles. Il est ainsi possible de tester le programme pour un nombre maximum de 12 prolongations. Pour un nombre au-delà de 12, le programme renverra un message d'erreur à l'utilisateur lui indiquant que le nombre est trop grand. Le nombre 12 a été choisi après de multiples tests et au vue de la longueur d'exécution du code. Ce nombre fournit un rapport efficacité / temps d'attente acceptable. Au-delà, le temps d'attente devient trop important.

```
x = int(input('Entrez le nombre de prolongation désiré : '))
nb_prolongation = [1,2,3,4,5,6,7,8,9,10,11,12]

if x in nb_prolongation:
```

```
    else: # Si jamais le nombre de prolongation n'est pas dans la liste, plusieurs cas de figure
        if x > 12: # Si le nombre est trop grand, il n'est pas dans la liste. Pour éviter d'attendre trop longtemps (ou de faire crasher le pc),
            print('Le nombre est trop grand ! Le maximum est 12 prolongations (pour éviter de faire crasher le pc).')
            return nbre_prolongation() # On relance directement le programme
        elif x == 0: # On distingue le cas ou l'utilisateur ne veut pas de prolongation

            score_equipe1=0 # On mets les scores à 0
            score_equipe2=0

            while score_equipe1 == score_equipe2: # Tant que les scores sont égaux, on rejoue un match de 40 min pour avoir un gagnant au bout d

                Match40min()
                #print(score_equipe1, score_equipe2) permet de voir que le programme recommence s'il y a égalité

            print('Score final :', score_equipe1, '-', score_equipe2)

            if score_equipe1 > score_equipe2: # On regarde qui à gagné le match
                print('L\'équipe 1 remporte le match')
            else: # On regarde qui à gagné le match
                 print('L\'équipe 2 remporte le match')

        else: # Si le nombre entré par l'utilisateur est <0 on lui indique que son nombre n'est pas valide
            print('Le nombre n\'est pas valide, recommencez !')
            return nbre_prolongation() # On relance directement le programme

#nbre_prolongation()
```

L'utilisateur donnera une valeur, stockée dans "x". Ensuite notre programme fonctionne de la façon suivante :

Tant que "i" est différent de "x" on regarde le compteur de prolongation ("i").
Si "i" = 0 alors on est dans le cas où il n'y a pas eu de prolongation, c'est-à-dire le match principal donc, on rejoue un match tant qu'il n'y a pas une égalité.
S'il y a égalité, alors on ajoute 1 au compteur de prolongation.
A partir de maintenant, on va jouer uniquement des prolongations (le match principal ne sera plus rejoué) jusqu'à arriver au nombre de l'utilisateur.
On enregistre chaque score d'une équipe dans sa liste respective (equipe 1 pour l'équipe 1 et equipe2 pour l'équipe 2). Si deux scores sont différents à la fin d'une prolongation, alors on recommence depuis le début.
Quand i = x, il faut jouer une dernière prolongation où le résultat donnera un vainqueur. Si ce n'est pas le cas, alors c'est qu'il y a encore égalité. Il faut donc tout recommencer.

```
while i != x: # Tant que le nombre de prolongations n'est pas égal à celui demandé par l'utilisateur la boucle tant que va s'éxecuter

    if i == 0: # Si le nombre de prolongation = 0, on n'a pas encore joué de match donc on joue le match de 40 min

        while score_equipe1 != score_equipe2: # Tant que les scores sont différents, on génère un nouveau match pour avoir une égalité

            equipe1 = [] # On reinitialise les deux listes à chaque fois qu'on recommence
            equipe2 = []

            Match40min()

        equipe1.append(score_equipe1) # On ajoute dans les listes qui stockent les scores les scores des deux équipes
        equipe2.append(score_equipe2)

        i+=1 # les deux équipes sont à égalités donc le nombre de prolongation augmente de 1

    if i > 0: # si le nombre de prolongation est >0 il faut jouer des prolongations

        while score_equipe1 == score_equipe2 and i!=x: # Tant que les scores sont égaux et que le nombre de prolongation n'est pas égal à celui demande

            prolongations() # On joue une prolongation

            if score_equipe1 == score_equipe2: # Si les scores sont égaux, on stocke les scores dans les listes associés aux deux équipes
                equipe1.append(score_equipe1)
                equipe2.append(score_equipe2)
                i+=1 # le nombre de prolongation augmente de 1

            else:
                i=0 # Si les scores sont différents, on ne respecte pas la condition du nombre de prolongation donc on met le nombre de prolongation a

    if i==x: # Si le nombre de prolongation est égal à celui demandé par l'utilisateur alors on joue une dernière prolongation pour avoir un vainqueur

        prolongations()

        if score_equipe1 != score_equipe2: # Si les scores sont différents, on les ajoute dans les listes qui stockent les scores
            equipe1.append(score_equipe1)
            equipe2.append(score_equipe2)
        else:
            i=0 # S'il y a encore égalité, on n'a pas le nombre désiré par l'utilisateur mais un nombre plus important, il faut tout recommencer
```

Une fois le bon nombre de prolongation atteint et tous les résultats stockés dans les listes, on compare les scores des équipes pour savoir qui a gagné. A ce stade, aucune égalité n'est possible, il y a forcément un vainqueur.
Grâce à une boucle for et pour faciliter la compréhension, nous affichons tous les scores et la prolongation à laquelle ils correspondent. Enfin nous finirons par donner le vainqueur et le résultat final du match !

```
for p in range(i+1): # La boucle sert à afficher les scores et le mot "PROLONGATION"
    if p == 0:
        print('\nScore match 40 min :', equipe1[0], '-', equipe2[0], '\n\nPROLONGATION 1',)
    elif p<i:
        print('Score :', equipe1[p], '-', equipe2[p], '\n\nPROLONGATION', p+1, '')
    else:
        if score_equipe1 > score_equipe2: # On regarde quelle équipe à gagné le match
            print('Score Final :', equipe1[p], '-', equipe2[p])
            print('\nL\'équipe 1 remporte le match')
        else: # On regarde quelle équipe à gagné le match
            print('Score Final :', equipe1[p], '-', equipe2[p])
            print('\nL\'équipe 2 remporte le match')
```

Finalement, pour laisser la possibilité à l'utilisateur de consulter la partie qu'il souhaite, nous avons créé la fonction "partie" suivante :

```
def partie():

    try: # On va demander à l'utilisateur une saisie clavier, on va faire en sorte que s'il n'entre pas quelque chose de valide le programme en ait conscience
        numero = int(input('\nQuelle partie voulez vous lancer ? '))
        if numero == 1: # Partie 1
            print('*'*57)
            print('La classe contient', len(s), 'étudiants') # Question 1
            print('*'*57)
            gender() # Question 2
            print('*'*57)
            popu() # Question 3
            print('*'*57)
            sport_pref() # Question 4
            print('*'*57)
            sport_pas_pref() # Question 4
            print('*'*57)
            satisfaction() # Question 5
            print('*'*57)
            r = input('Voulez vous regarder une autre partie ? (y/n) ') # On demande à l'utilisateur s'il veut relancer une partie
            if r == 'y': # Si son résultat est oui, on relance le programme partie
                partie()
            else: # Sinon, le programme se termine
                print('\nMerci d\'avoir regardé le programme !')

        elif numero == 2: # Partie 2
            print('*'*57)
            choix(97, 256, 181) # Question 1
            print('*'*57)
            print(menu.keys()) # Question 2 nous voulons tester si notre dictionnaire fonctionne bien, on va tester quelques commandes
            print(menu.values())
            print('Menu A :' ,menu['Menu A'])
            print('Menu B :' ,menu['Menu B'])
            print('Menu C :' ,menu['Menu C'])
            print('*'*57)
            r = input('Voulez vous regarder une autre partie ? (y/n) ')
            if r == 'y':
                partie()
            else:
                print('\nMerci d\'avoir regardé le programme !')

        elif numero == 3: # Partie 3
            print('*'*57)
            print('MATCH DE BASKET CLASSIQUE')
            Partie_Basket() # Question 1
            print('*'*57)
            print('MATCH DE BASKET AVEC STATISTIQUES') # Question 1 avec statistiques
            Stats_Match()
            print('*'*57)
            print('MATCH DE BASKET AVEC PROLONGATITIONS ET PROBABILITÉS CHANGÉES')
            Partie_Basket2() # Question 2
            print('*'*57)
            print('MATCH DE BASKET OU L\'ON DEMANDE LE NOMBRE DE POLONGATION(S) SOUHAITÉE(S)')
            nbre_prolongation() # Question 2 pour un nombre de polongation donné par l'utilisateur
            print('*'*57)
            r = input('Voulez vous regarder une autre partie ? (y/n) ')
            if r == 'y':
                partie()
            else:
                print('\nMerci d\'avoir regardé le programme !')

        else: # Nombre trop grand ou trop petit
            print('Mauvais nombre ! Les parties sont 1,2 ou 3')
            partie()
    except: # On rejette le cas ou l'utilisateur aurait entrée du texte à la place d'un nombre par exemple (il faut qu'il entre un nombre valide)
        print('Le nombre n\'est pas valide, il y a 3 parties dans ce devoir')
        partie() # On relance immédiatement le programme

partie()
```

Cette fonction permet à l'utilisateur, suivant le chiffre qu'il entre, de consulter la partie de son choix.
Si par inadvertance, il aurait entré un nombre trop petit ou trop grand, le programme lui rappellera qu'il n'y a que 3 parties dans ce sujet.
Si toujours par inadvertance, il avait appuyé accidentellement sur une ou plusieurs autres touches de son clavier, alors le programme lui indiquerait que sa saisie est erronée.

Après avoir consulté une partie, le programme demandera à l'utilisateur s'il souhaite continuer et accéder à une autre partie ou s'il préfère s'arrêter la. L'utilisateur doit répondre "y" s'il veut regarder une autre partie ou "n" s'il ne veut pas consulter une autre partie.