Séance 11 :
Probabilités

1. ***Donner les fonctions*** `espe(X)` ***et*** `varia(X)` ***qui donnent l'espérance et la variance d'une liste*** `X` ***donnant la loi d'une variable aléatoire à valeurs dans*** $\mathbb{N}$***.***

   On a, par définition de l'espérance :

$$E(X) = \sum_{i=0}^{+\infty} x_i \mathbb{P}(X = x_i)$$

   D'où :

```python
def espe(X):
    return sum([i*X[i] for i in range(len(X))])
```

   De même, par définition de la variance :

$$V(X) = E(X^2) - E(X)^2$$

   Or, d'après le théorème de transfert :

$$E(X^2) = \sum_{i=0}^{+\infty} i^2 \mathbb{P}(X = x_i)$$

   D'où :

```python
def varia(X):
    E2 = sum([i**2*X[i] for i in range(len(X))])
    E = espe(X) ** 2
    return (E2 - E)
```

2. ***Tester vos fonctions avec la loi d'un dé.***

   On sait qu'un dé suit une loi uniforme de paramètre $p = \frac{1}{6}$.
   Soit $X \hookrightarrow \mathcal{U}\left(\frac{1}{6}\right)$. Alors :

$$\begin{aligned} E(X) &= \tfrac{n+1}{2} = \tfrac{7}{2} \\ V(X) &= \tfrac{n^2-1}{12} = \tfrac{35}{12} \end{aligned}$$

   En exécutant

```python
X = [0] + [1/6]*6
print(espe(X))
print(varia(X))
```

   J'obtiens :

$$\begin{aligned} E(X) &= \texttt{3.5} \\ V(X) &= \texttt{2.916666666666666} \end{aligned}$$

3. ***Soit*** S_d ***la variable aléatoire donnant la loi de la somme de*** d ***dés, montrer que :***

$$\mathbb{P}(S_d = k) = \frac{1}{6} \sum_{j=1}^{6} \mathbb{P}(S_{d-1} = k - j)$$

   Soit Ω l'univers. Alors, $X(\Omega) = [\![d, 6d]\!]$.
   On pose Z la variable aléatoire donnant la loi du dernier dé.

Alors, $((Z = j))_{1 \leq j \leq 6}$ est un système complet d'évènements. Ainsi, pour un évenement E quelconque, la formule des probabilités totales donne :

$$\mathbb{P}(E) = \sum_{j=1}^{6} \mathbb{P}(E|Z = j)\mathbb{P}(Z = j)$$

Ainsi :

$$\begin{aligned} & \mathbb{P}(S_d = k) &=& \tfrac{1}{6} \sum_{j=1}^{6} \mathbb{P}(S_d = k|Z = j) \\ \Rightarrow\ & \mathbb{P}(S_d = k) &=& \tfrac{1}{6} \sum_{j=1}^{6} \mathbb{P}(S_{d-1} = k - j) \end{aligned}$$

On retrouve bien la formule précédente.

4. ***Construire la fonction*** `tcheb(d, p)` ***qui renvoie les lois de la somme de*** i ***dés avec*** $1 \leq i \leq d$ ***sous forme de tableau*** T ***tel que*** `T[i, k]` $= \mathbb{P}(S_i = k)$.
Construire `tcheb(d, p)` ***par récursivité en s'inspirant de la construction du triangle de Pascal.***

```
def tcheb(d, p):
    if (d == 1):
        return [[0] + [1/6] * 6 + [0] * (p-7)]
    t = tcheb(d-1, p)
    newline = [0] + [1/6*sum(t[-1][max(0, k-6):k]) for k in range(1, p)]
    t.append(newline)
    return t
```

5. ***Construire alors*** `S(d)` ***renvoyant la loi de la somme de*** d ***dés. Vérifier ainsi les valeurs de l'espérance et de la variance de la liste*** $S(d)$***.***

```
def S(d):
    return tcheb(d, 6*d+1)[1]
```

Je vérifie avec `d = 3` par exemple. Je devrais trouver :

$$\begin{aligned} E(S_d) &= d * \tfrac{7}{2} &=& 10,5 \\ V(S_d) &= d * \tfrac{35}{12} &=& 8,75 \end{aligned}$$

```
S2 = S(2)
print(espe(S2))
print(varia(S2))
```

Et je trouve :

$$\begin{aligned} E(S_d) &= \texttt{10.499999999999998} \\ V(S_d) &= \texttt{8.750000000000043} \end{aligned}$$

6. ***Tracer les graphes des trois fonctions de répartitions des variables*** $X_d = \dfrac{\texttt{S(d)}}{\texttt{espe(S(d))}}$ ***pour*** `d=5` ***en rouge,*** `d = 20` ***en vert et*** `d = 100` ***en bleu.***

❏ D'abord je crée la fonction `repartition` qui retourne les valeurs de la fonction répartition de la variable aléatoire `Z` pour les abscisses passée en paramètres dans la liste `T` :

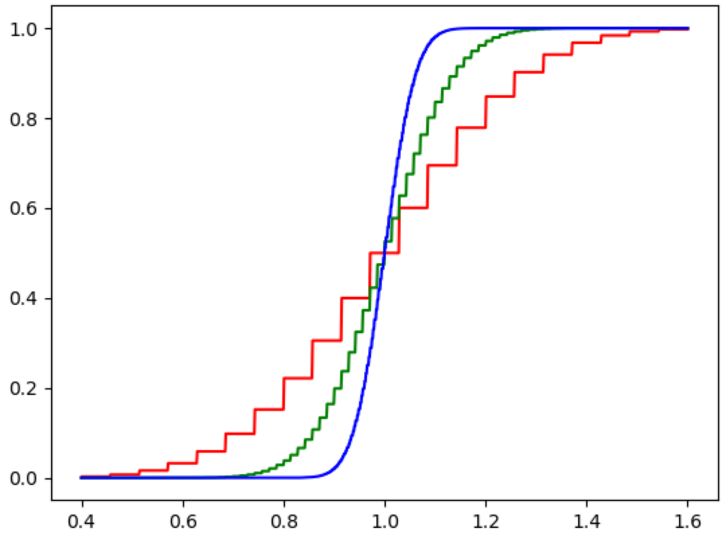
```
def repartition(Z, T):
    A = []
    for t in T:
        q = min(len(Z), int(t*espe(Z)) + 1)
        A.append(sum(Z[:q]))
    return A
```

❑ Ensuite je trace tout ça :

```
def TraceRepartition():
    X = np.linspace(0.4, 1.6, 1000)

    params = [
        (5, 'r'),
        (20, 'g'),
        (100, 'b')
    ]

    for d, color in params:
        Z = S(d)
        Y = repartition(Z, X)
        plt.plot(X, Y, color, label=str(d))
    plt.show()
```



(a) ***Que va devenir la fonction de répartition pour un grand nombre de dés ?*** Elle devient de plus en plus lisse. Pour d très grand elle ressemblera à :

$$f(x) = \begin{cases} 0 & \text{pour } x < 1 \\ 1 & \text{pour } x > 1 \end{cases}$$

(b) ***Expliquer ce résultat avec la variance et l'inégalité de Bienaymé-Tchebychev.*** Inégalité de Bienaymé-Tchebychev :

$$\mathbb{P}(|S_d - E(S_d)| \geq \alpha) \leq \frac{V(S_d)}{\alpha^2}$$

7. ***On s'intéresse à la loi expérimentale de la somme de trois dés.***
Construire une fonction `Des_3(N)` ***qui donne expérimentalement la loi de la somme de trois dés lorsqu'on lance*** N ***fois de suite trois dés ensemble. On pourra utiliser la fonction*** `randint(a,b)` ***de la bibliothèque*** `random`***.***
Pour $N = 8000$ ***lancers de trois dés, vérifier que la loi expérimentale présente une erreur d'environ 5% par rapport au maximum de la loi théorique.***

❏ Je crée d'abord la fonction `Des_3` :

```
def Des_3(N):
    X = [0] * (6*3+1)

    for i in range(N):
        # On lance les 3 des
        des = [rd.randint(1, 6) for k in range(3)]

        # Somme des 3 des
        k = sum(des)

        # On enregistre le resultat
        X[k] += 1/N
    return X
```

❏ Ensuite, pour `e` expériences, je fais `l` lancés, et j'enregistre les erreurs entre la loi théorique (`Th`) et la loi expérimentale (`Xp`). Ensuite je trace tout ça.

```
def TraceExp(e, l):
    X = [k for k in range(1, e + 1)]
    Y = []

    # Les valeurs theoriques
    Th = np.array(S(3))

    er = []
    # On fait e experiences
    for k in range(1, e+1):

        # Les valeurs experimentales
        Xp = np.array(Des_3(l))

        # Les erreurs
        dif = np.abs(Xp - Th)
        er.append(np.max(dif)/np.max(Th))

        # On rajoute la moyenne des erreurs
        Y.append(sum(er)/len(er))
    plt.plot(X, Y, 'rx')
    plt.show()

    return (max(Y))
```

En exécutant `print(TraceExp(50, 8000) * 100)`, j'obtiens : e_{max} = `4.823192239858807`%.