TP4 : Fonction `rd.random()` et simulation de v.a.r. discrètes suivant des lois usuelles

Commencer par importer les bibliothèques suivantes dans chaque fichier **Python** utilisé :

```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
```

I. La fonction `rd.random()`

I.1. Que fait-elle ?

► Entrer une dizaine de fois dans la console la commande `rd.random()` (on peut utiliser la flèche du haut pour faire réapparaitre la commande précédemment entrée dans la console). Qu'observe-t-on ?

► Entrer dans la console la commande `rd.random(10)`. Que renvoie la commande `rd.random(d)` ?

► Recopier le script suivant et le compiler. Comment sont obtenus les tracés ? Quel phénomène voit-on apparaître ?

```
for k in range(1,7):
    T = rd.random(10**k)
    plt.hist(T)
    plt.title(f'Répartition de {10**k} nombres générés par la fonction
              rd.random()')
    plt.show()
```

- On modifie légèrement le script précédent :

```
for k in range(1,7):
    T = rd.random(10**k)
    plt.hist(T, density = True)
    plt.title(f'Répartition normalisée de {10**k} nombres générés par la
              fonction rd.random()')
    plt.show()
```

Quelle est la différence entre les nouveaux tracés et les anciens ?

- La commande `rd.random()` permet de simuler quelle v.a.r. ?

I.2. A quoi sert-elle ?

- Exécuter une dizaine de fois le script suivant. Qu'observe-t-on ?

```
r = rd.random()
print(r)
print(r < 1/3)
```

- Quelle est la probabilité que **Python** renvoie `True` lorsqu'on exécute la commande `rd.random() < 1/3` ?

- En déduire une commande qui renvoie `True` avec probabilité p (le réel $p \in]0,1[$ étant fixé). Cette commande permet alors de simuler une épreuve de Bernoulli de paramètre p.

II. Simulation de v.a.r. suivant une loi de Bernoulli

On rappelle qu'une épreuve de Bernoulli est une expérience aléatoire à deux issues : `Succès` et `Echec`. Si X est la v.a.r. qui vaut 1 en cas de `Succès` et 0 en cas d'`Echec`, alors X suit une loi de Bernoulli. Plus précisément, si la probabilité de `Succès` est p, alors X suit la loi de Bernoulli de paramètre p. On le note : $X \hookrightarrow \mathcal{B}(p)$.

- Compléter la fonction suivante pour qu'elle
  - prenne en argument un réel $p \in]0,1[$
  - renvoie une simulation d'une v.a.r. $X \hookrightarrow \mathcal{B}(p)$

```
def Bernoulli(p):
    if rd.random() < p:
        return _____
    else:
        return _____
```

- Compléter la fonction suivante pour qu'elle
  - prenne en arguments un entier $N \in \mathbb{N}^*$ et un réel $p \in]0,1[$
  - renvoie une liste contenant le résultat de la simulation de N v.a.r. indépendantes suivant toutes la loi $\mathcal{B}(p)$

```
def SampleBernoulli(N,p):
    L = ______
    for k in _____________:
        ____________________________
    return L
```

- Recopier et exécuter le script suivant :

```
N = 10**4
p = 0.3
echantillon = SampleBernoulli(N,p)
effectif = np.arange(0,2)
for k in echantillon:
    effectif[k] += 1
plt.bar([0,1], effectif)
plt.title(f'Répartition des valeurs dans l\'échantillon de {N}
        var suivant la loi B(p)')
plt.show()
```

Interpréter les graphiques obtenus :

III. Simulation de v.a.r. suivant une loi binomiale

On rappelle que lors d'une répétition de n épreuves de Bernoulli identiques et indépendantes, la v.a.r. X qui compte le nombre de succès suit une loi binomiale. Plus précisément, si la probabilité de succès de chaque épreuve de Bernoulli est p, alors X suit la loi binomiale de paramètres n et p. On le note : $X \hookrightarrow \mathcal{B}(n,p)$.
On rappelle également que si $X_1, \dots, X_n$ sont des v.a.r. qui sont indépendantes et suivent toutes la loi $\mathcal{B}(p)$, alors la v.a.r. $X = X_1 + \cdots + X_n$ suit la loi $\mathcal{B}(n,p)$.

- En s'inspirant de la remarque précédente, compléter la fonction suivante pour qu'elle
  - prenne en arguments un entier $n \in \mathbb{N}^*$ et un réel $p \in]0,1[$
  - renvoie une simulation d'une v.a.r. $X \hookrightarrow \mathcal{B}(n,p)$

```
def Binomiale(n,p):
    X = 0
    for k in _____________:
        if ____________________:
            X = _________
    return _____
```

- Compléter la fonction suivante pour qu'elle
  - prenne en arguments un entier $N \in \mathbb{N}^*$, un entier $n \in \mathbb{N}^*$ et un réel $p \in]0,1[$
  - renvoie une liste contenant le résultat de la simulation de N v.a.r. indépendantes suivant toutes la loi $\mathcal{B}(n,p)$

```
def SampleBinomiale(N,n,p):
    L = ______
    for k in _____________:
        ___________________________________
    return L
```

- Recopier et compiler le script suivant :

```
N = 10000
n = 10
for k in range(1,10):
    p = k/10
    echantillon = SampleBinomiale(N,n,p)
    effectif = np.arange(n+1)
    for k in echantillon:
        effectif[k] += 1
    plt.bar(np.arange(n+1), effectif)
    plt.title(f'Répartition des valeurs dans l\'échantillon de N var
                suivant la loi B(n,p)')
    plt.show()
```

Interpréter les graphiques obtenus :

▶ Faire un copié collé du script précédent et le modifier comme ci-dessous :

```
N = 10000
n = 10
for k in range(1,10):
    p = k/10
    echantillon = SampleBinomiale(N,n,p)
    effectif = np.arange(n+1)
    for k in echantillon:
        effectif[k] += 1
    frequence = effectif / N
    plt.bar(np.arange(n+1), frequence)
    plt.title(f'Répartition normalisée des valeurs dans l\'échantillon de {N} var
              suivant la loi B({n},{p})')
    L = [np.math.comb(10,k)*p**k*(1-p)**(10-k) for k in range(11)]
    plt.plot(L,'+r')
    plt.show()
```

▶ Que contient la liste `L` ?

▶ Interpréter les graphiques obtenus :

IV. Simulation de v.a.r. suivant une loi géométrique

On rappelle que lors d'une répétition *infinie* d'épreuves de Bernoulli identiques et indépendantes, la v.a.r. X égale au rang du premier succès suit une loi géométrique. Plus précisément, si la probabilité de succès de chaque épreuve de Bernoulli est p, alors X suit la loi géométrique de paramètre p. On le note : $X \hookrightarrow \mathcal{G}(p)$.

▶ Compléter la fonction suivante pour qu'elle

- prenne en argument un réel $p \in]0,1[$
- renvoie une simulation d'une v.a.r. $X \hookrightarrow \mathcal{G}(p)$

```
def Geometrique(p):
    X = 1
    while ____________________:
        X = _________
    return _____
```

- Est-on sûr que cette fonction s'arrête toujours ?

- Compléter la fonction suivante pour qu'elle
  - prenne en arguments un entier $N \in \mathbb{N}^*$ et un réel $p \in]0,1[$
  - renvoie une liste contenant le résultat de la simulation de N v.a.r. indépendantes suivant toutes la loi $\mathcal{G}(p)$

```
def SampleGeometrique(N,p):
    L = ______
    for k in _____________:
        ___________________________________
    return L
```

- Recopier et compiler le script suivant :

```
N = 100000
p = 0.3
echantillon = SampleGeometrique(N,p)
effectif = np.arange(1,20)
for k in echantillon:
    if k < 20:
        effectif[k-1] += 1
frequence = effectif / N
plt.bar(np.arange(1,20), frequence)
plt.title(f'Répartition normalisée des valeurs dans l\'échantillon de {N} var
          suivant la loi G({p})')
A = [k for k in range(1,20)]
L = [p*(1-p)**(k-1) for k in range(1,20)]
plt.plot(A,L,'+r')
plt.show()
```

Interpréter les graphiques obtenus :

V. Simulation de v.a.r. suivant une loi uniforme discrète

- On considère deux entiers naturels a et b tels que $a \leqslant b$. Décrire ce que renvoient les commandes suivantes :
  - `rd.random()`
  - `(b-a+1)*rd.random()`
  - `np.floor((b-a+1)*rd.random())`
  - `a + np.floor((b-a+1)*rd.random())`

- Compléter la fonction suivante pour qu'elle
  - prenne en arguments deux entiers naturels a et b tels que $a \leqslant b$
  - renvoie une simulation d'une v.a.r. $X \hookrightarrow \mathcal{U}(\llbracket a, b \rrbracket)$

```
1  def Uniforme(a,b):
2      return ____________________________________________
```

- Compléter la fonction suivante pour qu'elle
  - prenne en arguments un entier $N \in \mathbb{N}^*$ et deux entiers naturels a et b tels que $a \leqslant b$
  - renvoie une liste contenant le résultat de la simulation de N v.a.r. indépendantes suivant toutes la loi $\mathcal{U}(\llbracket a, b \rrbracket)$

```
1  def SampleUniforme(N,a,b):
2      L = ______
3      for k in _____________:
4          ___________________________________
5      return L
```

- Recopier et compiler le script suivant :

```
for k in range(1,7):
    N = 10**k
    a = 0
    b = 4
    echantillon = SampleUniforme(N,a,b)
    effectif = np.arange(a, b+1)
    for k in echantillon:
        effectif[k-1] += 1
    plt.bar(np.arange(a, b+1), effectif)
    plt.title(f'Répartition des valeurs dans l\'échantillon de {N} var
              suivant la loi U([|{a},{b}|])')
    plt.show()
```

Interpréter les graphiques obtenus :

Commentaire

Avoir en tête les diagrammes à bâtons théoriques des lois usuelles permet d'émettre des conjectures sur la loi suivie par une variable aléatoire inconnue, que l'on sait simuler, à partir du diagramme à bâtons (ou de l'histogramme) des fréquences obtenues sur un grand échantillon de la variable.