Simulation de variables aléatoires discrètes

L'objet de ce TD est de programmer des fonctions qui simulent le comportement d'une variable aléatoire discrètes. Les méthodes sont notablement différentes, selon que l'on veuille simuler une variable discrète ou une variable continue. Néanmoins, elles utilisent toutes la fonction `random.random()` qui simule une variable aléatoire de loi uniforme sur $[\,0\,;1\,[$.

Ex. 1 — Processus de Bernoulli

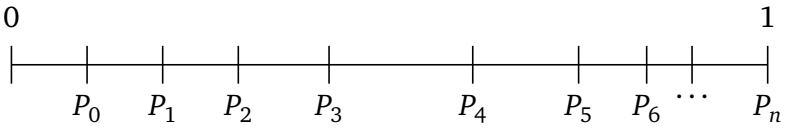
1. Proposer et implémenter une fonction `Bernoulli(p)` qui simule une variable de Bernoulli de paramètre p.
2. Une v.a.r. de loi binomiale $\mathscr{B}(n,p)$ peut être vue comme la somme de n v.a.r. de Bernoulli indépendantes et de loi $\mathscr{B}(1,p)$. Proposer et implémenter une fonction `Binomiale(n, p)` qui simule une variable de loi binomiale $\mathscr{B}(n,p)$.
   Faire 1000 simulations et en tracer l'histogramme à l'aide de la fonction `hist` de la librairie `matplotlib.pyplot`
3. Rappeler l'expérience-type d'une loi géométrique $\mathscr{G}(p)$.
   Proposer et implémenter une fonction `geometrique(p)` qui simule une variable de loi géométrique $\mathscr{G}(p)$.
4. Recopier et modifier la fonction précédente pour simuler la loi $\mathscr{T}(n,p)$ du temps d'attente du n–ième succès dans un processus de Bernoulli.

Ex. 2 — Une méthode plus générale Cette méthode s'applique aussi bien aux v.a.r. discrètes à support fini qu'infini.
Soit une loi discrète sur $\mathbb{N}$ définie par la donnée de réels $(p_k)_{k\in\mathbb{N}}$. On pose

$$\forall k \in [\![\,1\,;n\,]\!], \qquad P_k = \sum_{i=0}^{k} p_i.$$

Soit U une variable aléatoire de loi $\mathscr{U}\big([\,0\,;1\,]\big)$ et X la variable aléatoire définie par $X = \inf\{k\in\mathbb{N},\quad U \leqslant P_k\}$.

1. Démontrer que X suit la loi définie par les réels $(p_k)_{k\in\mathbb{N}}$. On pourra s'aider du schéma ci-dessous



2. On suppose que les coefficients de la loi de X sont renvoyés par une fonction `loiX(k)`. Expliquer le fonctionnement de la fonction suivante

```
from random import random

def Simule_LoiFinie(loiX):
    U = random()
    Cumul =loiX(0)
    k = 0

    while U > Cumul:
        k = k + 1
        Cumul = Cumul + loiX(k)
    return k
```

3. *Loi de Poisson* Afin d'éviter le calcul de factorielles (toujours très coûteux), on va adapter la fonction suivante. Rappeler la définition des coefficients p_k de la loi de Poisson $\mathscr{P}(\lambda)$ et montrer que

$$p_0 = P_0 = \mathrm{e}^{-\lambda} \quad \forall k\in\mathbb{N}^*, \quad p_k = \frac{\lambda}{k}p_{k-1} \quad P_k = P_{k-1} + p_k$$

   Proposer une modification de la fonction afin de simuler la loi de Poisson $\mathscr{P}(\lambda)$.

Ex. 3 — Processus de Galton–Watson

On s'intéresse à la descendance d'un individu, qui constitue la génération 0. À chaque génération, un individu donne naissance à des enfants et disparaît. Le nombre de descendants des différents individus sont des v.a.r. indépendantes et de même loi. Cette loi est supposée d'espérance finie μ.

1. Écrire une fonction `GW_Poisson(n, l)` qui renvoie le nombre d'individu aux générations 1 à n, avec comme loi de reproduction une loi de Poisson de paramètre `l`.
2. Représenter, en fonction du temps et sur un même graphe, l'évolution de la moyenne du nombre d'individus sur 20 simulations. Faire varier `l` ; que remarquez-vous ?
3. Représenter le graphe de la probabilité d'extinction de la descendance d'un individu en fonction de μ.
4. Reprendre la question précédente avec une loi géométrique sur $\mathbb{N}$ de paramètre p.