Python td 5 - Simulation de lois discrètes

Les générateurs de nombres aléatoires en Python créent des **matrices** dont les cases sont des simulations de variables suivant des lois classiques.

```
import numpy.random as rd
```

I. Générateur `rd.random`

L'instruction `x=rd.random()` renvoie un réel pris au hasard entre 0 et 1. Autrement dit, il s'agit de la simulation d'une variable X suivant la loi uniforme **continue** $X \hookrightarrow \mathcal{U}([0,1])$ (voir le cours sur les variables à densité).

L'instruction `L=rd.random(n)` renvoie une matrice ligne L à n cases (matrice unidimensionnelle), chaque coefficient de cette matrice prenant pour valeur un réel au hasard entre 0 et 1.

L'instruction `A=rd.random([n,p])` renvoie une matrice rectangulaire A à n lignes et p colonnes dont chaque coefficient prend pour valeur un réel au hasard entre 0 et 1.

Exercice 1
Quelle est la loi suivie par la variable aléatoire X définie dans la fonction suivante ?

```
def X(p):
    a=rd.random()
    if a<p:
          y=1
    else:
          y=0
    return y
```

II. Loi uniforme discrète

Etant donnés deux nombres entiers a et b, avec $a < b$, l'instruction `A=rd.randint(a,b+1)` renvoie un entier pris au hasard dans $[[a,b]]$, c'est-à-dire est une simulation d'une variable aléatoire X suivant la loi $\mathcal{U}([[a,b]])$.
Attention la dernière borne n'est pas atteinte !!.

L'instruction `L=rd.randint(a,b+1,n)` renvoie une matrice ligne L à n cases, chaque case étant un entier pris au hasard dans $[[a,b]]$.
L'instruction `A=rd.randint(a,b+1,[n,p])` renvoie une matrice rectangulaire a à n lignes et p colonnes, chaque coefficient étant encore un entier pris au hasard dans $[[a,b]]$.

Exercice 2
Ecrire un petit programme qui donne soit 1, soit -1, et ceci de façon équiprobable.

Exercice 3
Une première simulation
Ecrire un programme qui simule 100 lancers d'un dé à six faces équilibré.

1. Créer une matrice L contenant les résultats de 1000 lancers d'un dé à six faces équilibré.

   On souhaite compter le nombre de fois où le chiffre 6 est apparu au cours des 1000 lancers.

2. Méthode 1 : à l'aide d'une boucle `for`, parcourir les cases de la matrice et compter le nombre de fois où le chiffre 6 est sorti.

3. Méthode 2 :
   Taper `print(L==6)` et interpréter le résultat obtenu.
   Sachant qu'en Python, False correspond à 0 et True à 1, en déduire une méthode rapide pour compter le nombre de fois où le chiffre 6 est sorti.

Exercice 4
On considère l'expérience aléatoire suivante : on lance un dé équilibré jusqu'à ce que la somme des résultats obtenus soit supérieure ou égale à 11. On note alors X le nombre de lancers effectués. Ecrire un programme qui simule la variable aléatoire X et affiche le résultat.

III. Loi binômiale

III.1) Simulation à l'aide d'une loi de Bernoulli

Soit Y une variable aléatoire suivant la loi binômiale de paramètres (n,p). Y est alors égale au nombre de succès lorsque l'on répète n fois une épreuve de Bernoulli $\mathcal{E}$ ayant pour probabilité de succès p, de façon identique et indépendante.

A l'aide de la fonction `X` définie dans l'Exercice 1, écrire alors un programme qui demande n et p à l'utilisateur et qui simule Y.

III.2) A l'aide du générateur

La commande `a=rd.binomial(N,p)` simule une variable aléatoire suivant la loi binômiale $\mathcal{B}(N,p)$.

Comme pour chaque commande, on peut simuler avec la même instruction une matrice ligne L ou une matrice rectangulaire A dont les coefficients sont des simulations de cette loi.

Exemple : taper `A=rd.binomial(100,0.5,[3,5])`, puis `A=rd.binomial(100,0.9,[3,5])`, puis `A=rd.binomial(100,0.1,[3,5])`
Que constate-t-on ?

III.3) Représentation par un histogramme

```
import matplotlib.pyplot as plt
```

On simule 10000 fois la loi binômiale $\mathcal{B}(50, 0.5)$ via `L=rd.binomial(50,0.5,10000)`
Ensuite l'instruction `l=np.arange(0,51)` crée la liste de tous les entiers de 0 à 50.

Enfin les instructions `plt.hist(L,l)`
`plt.show()`
trace l'histogramme des valeurs prises par L en les répartissant dans les classes $[0, 1[$, $[1, 2[$, ..., $[49, 50]$
(la dernière classe inclus la valeur de droite).

Pour un résultat un peu plus joli :
`plt.clf()` (on efface la figure précédente)
`plt.hist(L,l,color = 'yellow',edgecolor = 'red')`
`plt.title("Simulations de la loi binômiale")`
`plt.show()`

Les paramètres de la fonction `plt.hist` devront être rappelés

On peut aussi spécifier le nombre de classes : ainsi `plt.hist(L,20)` répartirait les données dans 20 classes de même longueur.

IV. Simulation de la loi géométrique

IV.1) A partir de la loi de Bernoulli

Soit Z une variable suivant la loi géométrique de paramètre p. Z est alors égale au temps d'attente du premier succès lorsque l'on répète une épreuve de Bernoulli, de probabilité de succès p, de façon identique et indépendante.

Ecrire un programme qui demande p à l'utilisateur, puis qui simule Z en utilisant la fonction `X` définie dans l'Exercice 1.

IV.2) A l'aide du générateur

La commande `A=rd.geometric(p)` simule une variable aléatoire suivant **une loi géométrique de paramètre** p.
On peut également créer une matrice ligne, une matrice rectangulaire dont les coefficients sont des simulations de la loi géométrique.

Exercice 5
Ecrire un programme qui simule 1000 fois une variable X suivant la loi géométrique de paramètre $p = 0.1$ et qui représente les résultats obtenus sous la forme d'un histogramme.

V. Simulation de la loi de Poisson

Rappelons que la loi de Poisson ne correspond pas à un modèle concret.
La commande `A=rd.poisson(lambda)` simule une variable aléatoire suivant **une loi de Poisson de paramètre** `lambda`.

Exercice 6
Cinq professeurs s'occupent du stand "prépas ECG" du Salon de l'Etudiant. On suppose que le nombre de visiteurs par heure au stand prépa ECG du salon de l'étudiant suit une loi de Poisson de paramètre $p = 25$.

- Si le nombre de visiteurs est inférieur strictement à 20, deux des professeurs ont le temps d'aller boire un café.
- Si le nombre de visiteurs est compris au sens large entre 20 et 25, un seul professeur a le temps d'aller prendre un café.
- Enfin, si le nombre de visiteurs est strictement plus grand que 25, personne n'a le temps de prendre une pause.

Soit X la variable aléatoire égale au nombre de professeurs ayant pris un café pendant une heure donnée. Ecrire un programme qui simule la variable X et affiche le résultat.

VI. Diverses simulations

Exercice 7
Le problème du collectionneur d'images (cf **sujet Maths 2 2022** !!)

Soit $n \in \mathbb{N}^*$. En vue de la coupe d'Europe 2024, un enfant collectionne des images de foot. Pour compléter son album, il achète des paquets d'images jusqu'à ce qu'il possède les n images différentes de l'album. On suppose que chaque paquet contient une seule image, et que cette image est l'une des n possibles de facon équiprobable. Ecrire une fonction intitulée `def Collection(n) :` qui simule cette expérience et retourne le nombre total de paquets que l'enfant a dû acheter pour avoir la collection complète.

Exercice 8
Marche aléatoire le long d'un axe

Soit $p \in]0,1[$ et $n \in \mathbb{N}^*$. On considère une puce qui se déplace le long d'un axe gradué. Elle se trouve au départ à l'origine. Puis elle effectue à chaque fois un saut d'une unité vers la droite avec la probabilité p, ou d'une unité vers la gauche avec la probabilité $q = 1 - p$. On souhaite simuler n sauts successifs de la puce, les sauts étant supposés indépendants. Soit X_n la variable aléatoire égale à la position de la puce après le n-ième saut, et R_n le nombre de retours à l'origine au cours des n premiers sauts.

1. **Version 1 :** compléter le programme suivant pour simuler l'expérience, afficher les abscisses successives de la puce et le nombre de retours à l'origine après n sauts.

```
import numpy.random as rd
p=float(input("Entrer p avec 0<p<1 :")) #on demande p
n=int(input("Entrer un entier n>0 :")) #on demande n
x=0   #abscisse initiale
r=0   #nombre de retours à l'origine
for k in range(1,...):
   a=rd. ....(        )
   if a ...... :
         x=
   else:
         x=
   print(x)
   if x  .......:
         r=.........
print("r=",r)
```

2. **Version 2 :**

   (a) Soit B une VAR où $B \hookrightarrow \mathcal{B}(p)$ et $A = 2*B-1$. Quelle est la loi de A ? Justifier.

   (b) Compléter et comprendre le script suivant afin de simuler l'expérience, de représenter graphiquement les différentes situations de la puce, et d'afficher le nombre de retours à l'origine au cours des n premiers sauts.

```
import matplotlib.pyplot as plt
import numpy.random as rd
import numpy as np
p=float(input("p="))
n=int(input("Entrer un entier n>0 :"))
x=np.zeros(n+1)
x[0]=...   #abscisse initiale
for k in range(1,n+1):
    a=2*rd.binomial(....)-1
    x[k]=x[.....]+.....
print(x)
t=np.arange(0,n+1,1)
r=sum(x==0)-1
print("r=",r)
plt.plot(t,x)
plt.show()
```


   (c) Décrire le script suivant :

```
#déplacements sur une droite
p=float(input("p="))
n=int(input("Entrer un entier n>0 :"))
L=2*rd.binomial(1,p,n)-1
x=np.cumsum(L)
t=np.arange(1,n+1,1)
plt.plot(t,x)
plt.show()
r=np.sum(x==0)
print("r=",r)
```

Exercice 9
Marche aléatoire plane.

La puce se déplace cette fois-ci dans le plan. Elle se trouve au départ sur le point de coordonnées $(0,0)$. Puis à chaque saut, elle se déplace soit d'une unité vers le haut, soit d'une unité vers le bas, soit d'une unité vers la gauche, soit d'une unité vers la droite, et ceci de façon équiprobable. On souhaite simuler n sauts successifs de la puce ($n \in \mathbb{N}^*$) et représenter graphiquement ses déplacements.

1. Ecrire un programme qui simule n sauts de la puce et qui crée deux matrices lignes X et Y à n+1 cases, de sorte que $X[0] = Y[0] = 0$ et que la case $X[k]$ représente l'abscisse de la puce après k sauts et $Y[k]$ représente l'ordonnée de la puce après k sauts.

2. Représenter graphiquement les déplacements de la puce dans le plan dans le cas de 1000 sauts, puis de 10000, puis de 50000 sauts.
   On obtient une "fractale aléatoire"... c'est beau !