TP 20 - Phénomènes aléatoires

Les fonctions

Le module numpy contient un sous-module random :

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

On considère trois fonctions :

Notation	Explication
`rand()`	nombre aléatoire sur $[0,1[$
`rand(n)`	tableau ligne de n nombres aléatoires sur $[0,1[$
`rand(n,m)`	tableau de n lignes et m colonnes de nombres aléatoires sur $[0,1[$
`rand(u)`	tableau de nombres aléatoires sur $[0,1[$ dont la taille est celle du uplet u
`randint(n)`	entier aléatoire sur $[\![0,n[\![$
`randint(n,m)`	entier aléatoire sur $[\![n,m[\![$
	argument optionnel size=k pour un tableau ligne de k entiers et size=(i,j) pour un tableau de taille (i,j)
`choice(E)`	élément aléatoire de l'ensemble E
	argument optionnel size et replace=True (tirage avec remise par défaut) et p=[...] une loi de probabilité associée à E

Remarque : $\mathbf{P}(\texttt{rd.rand()} < p) = p$:

```
>>> a=rd.rand(4);a
array([ 0.54219811,  0.54660217,
0.01908569,  0.81227162])
>>> import numpy as np
>>> T=a>(0.5*np.ones(np.shape(a)));T
array([ True,  True, False,
True], dtype=bool)
>>> np.sum(T)
3
```

Tableau de 5 entiers aléatoires sur [0,12]

```
>>> [round(12*rd.rand()) for i in range(5)]
[7, 11, 10, 8, 3]
>>> rd.randint(0,13,size=5)
array([0, 1, 9, 2, 5])
>>> [int(13*rd.rand()) for i in range(5)]
[0, 3, 0, 5, 9]
```

Permutation aléatoire

```
>>> rd.choice(range(1,8),size=7,replace=False)
array([5, 4, 7, 3, 6, 2, 1])
```

Graine de l'aléa

L'instruction seed( ) permet de réinitialiser l'aléa : ce dernier est construit à l'aide d'une suite récurrente. Ainsi, l'aléa n'a que l'apparence de l'aléatoire car il est entièrement déterminé par la graine.

```
>>> rd.seed(123)
>>> rd.rand()
0.6964691855978616
>>> rd.rand()
0.28613933495037946
>>> rd.seed(123)
>>> rd.rand()
0.6964691855978616
```

Les autres instructions sont définit sur le même principe :

```
>>> rd.seed(12)
>>> [rd.randint(12) for i in range(10)]
[11, 11, 6, 1, 2, 3, 3, 0, 6, 1]
>>> rd.randint(18)
5
>>> rd.seed(12)
>>> rd.randint(12)
11
>>> [rd.randint(12) for i in range(9)]
[11, 6, 1, 2, 3, 3, 0, 6, 1]
```

Lois usuelles

Notation	Explication
`randint(a,b)`	simule $\mathscr{U}([\![a,b[\![)$, argument optionnel size
`binomial(n,p)`	simule $\mathscr{B}(n,p)$, argument optionnel size
`geometric(p)`	simule $\mathscr{G}(p)$, argument optionnel size
`poisson(l)`	simule $\mathscr{P}(l)$, argument optionnel size

```
>>> rd.binomial(6,0.3,size=(2,5))
array([[1, 2, 2, 3, 1],
       [2, 1, 4, 2, 5]])
```

Représentation de données

Le représentation de la loi ou de la fonction de répartition associée à une variable aléatoire peut se faire à l'aide des deux instructions suivantes du module graphique matplotlib.pyplot :

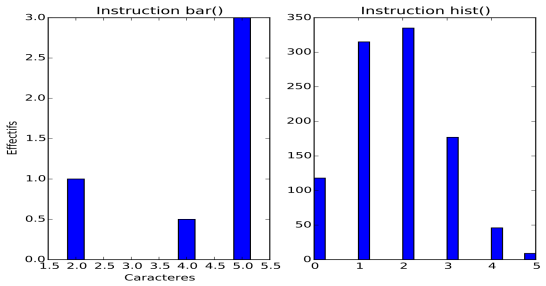
```
>>> import matplotlib.pyplot as plt
```

Notation	Explication
`bar(x,y)`	Trace les bâtons centrés aux oint de `x` et de hauteur les valeurs de `y`. Les deux liste sont de même taille. Les options pratiques : `width=0.3` épaisseur des colonnes `align='center'` positionnement `color`, `label`, ...
`hist(L,n)`	Trace l'histogramme des valeurs de la liste `L` réparties en n colonnes : `hist(L,bins=n)` On peut aussi imposer une subdivision : `bins=range(10)` ou `bins=[-2,0,1,5]`

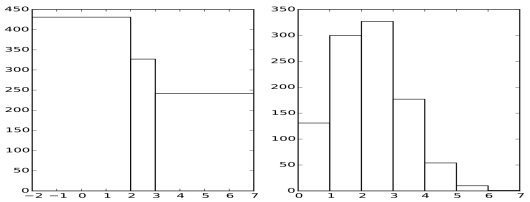
- Exemples d'utilisations :

```
plt.subplot(121)
plt.bar([2,4,5],[1,0.5,3],width=0.3,align='center')
plt.xlabel('Caracteres')
plt.ylabel('Effectifs')
plt.title('Instruction bar()')

plt.subplot(122)
a=rd.binomial(6,0.3,1000)
plt.hist(a,20)
plt.title('Instruction hist()')
```




```
a=rd.binomial(6,0.3,1000)
plt.subplot(121)
plt.hist(a,bins=[-2,2,3,7],color='white',ec='black')
plt.subplot(122)
plt.hist(a,bins=range(8),color='white',ec='black')
```



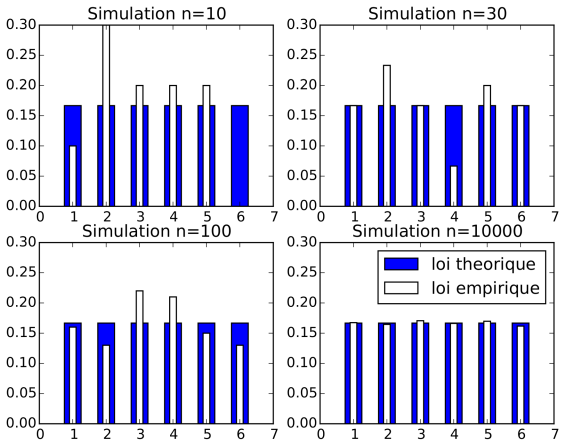
- Exemple illustrant la comvergence de la loi empirique vers la loi théorique lors d'une suite de lancers d'un dé :

Loi empirique

```
def de(n):
    T=rd.choice(range(0,6),n)
    P=6*[0]
    for e in T:
        P[e]=P[e]+1
    return([u/n for u in P])
```

Courbes

```
N=[10,30,100,10000]
x=range(1,7)
plt.close()
for i in range(4):
 plt.subplot(2,2,i+1)
 plt.ylim(0,0.3)
 plt.title('Simulation n='+str(N[i]))
 plt.bar(x,6*[1/6],width=0.5,label='loi theorique'
        ,align='center')
 plt.bar(x,de(N[i]),width=0.2,color='w',ec='black'
        ,align='center',label='loi empirique')
plt.legend(loc='best')
plt.show()
```



Exercices

Exercice 1 *Simulation de lois usuelles* Redéfinir les générateurs de loi usuelles en utilisant uniquement le générateur `rand()` :

- loi de Bernoulli $\mathscr{B}(p)$: `Ber(p)`

  choisir un réel au hasard dans $[0,1[$, renvoyer 1 si $x < p$ et 0 sinon.

- loi binomiale $\mathscr{B}(n,p)$: `bin(n,p)`

  renvoyer le nombre de succès, obtenus lors de n expériences `Bernoulli(p)` indépendantes.

- loi uniforme $\mathscr{U}(\llbracket a,b \rrbracket)$: `unif(a,b)`

  renvoyer un entier de $\llbracket a,b \rrbracket$ avec équiprobabilité.

- loi géométrique $\mathscr{G}(p)$: `geom(p)`

  renvoyer le rang du premier succès lors d'une suite d'expériences `Bernoulli(p)` indépendantes.

- loi hypergéométrique $\mathscr{G}(p)$: `hypergeo(N,n,p)`

  renvoyer le nombre de succès, obtenus lors de n tirages sans remise dans un ensemble de cardinal N contenant une proportion p d'éléments gagnants.

Réécrire chacun des lois précédentes (exceptée la loi géométrique) en utilisant la générateur `choice`.

Attention ! Pour simuler des lois usuelles, nous utiliserons maintenant les fonctions spécifiques du module `random`.

Exercice 2 Un joueur réalise une suite de lancers indépendants d'une pièce. Cette pièce donne pile avec la probabilité p ($0 < p < 1$). On note N la variable aléatoire égale au rang d'apparition du premier pile. Si N prend la valeur n, le joueur place n boules numérotées de 1 à n dans une urne, puis il extrait une boule au hasard de cette urne. On dit que le joueur a gagné si le numéro porté par la boule tirée est impair. On appelle X la variable aléatoire égale au numéro porté par la boule extraite de l'urne.

1. Compléter le script suivant qui simule cette expérience :

```
def experience(p):
    N=1
    while ...
        N=...
    X=...
    if ...
        print('Perdu',N,X)
    else:
        print('Gagne',N,X)
```

2. On rappelle que la loi faible des grands nombres assure que la fréquence empirique converge vers la probabilité de l'évènement.
Plus précisément, pour une suite (X_i) de variables aléatoires deux à deux indépendantes, de même loi et admettant un moment d'ordre 2, alors notant m l'espérance, pour tout $\varepsilon > 0$,

$$\mathbf{P}\left(\left|\frac{1}{n}\sum_{k=1}^{n} X_k - m\right| > \varepsilon\right) \underset{n\to+\infty}{\longrightarrow} 0$$

Ici, notant A l'évènement "le joueur gagne" alors

$$\forall i \in \mathbb{N}^*, \quad X_i = \mathbb{1}_A \hookrightarrow \mathscr{B}(\mathbf{P}(A))$$

D'où $\dfrac{1}{n}\displaystyle\sum_{k=1}^{n} X_k \to \mathbf{P}(A)$.
Donner une fonction `lfgn(p,nb)` qui donne une estimation de la probabilité que le joueur gagne.
On justifiera *rapidement* que $\mathbf{P}(A) > \dfrac{1}{2}$.

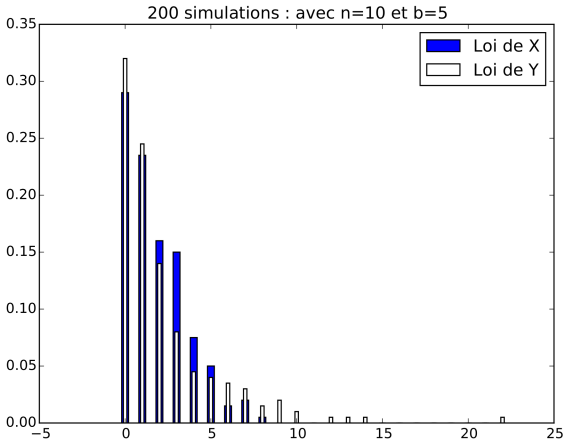
Exercice 3
1. Écrire le script d'une fonction, `Y()`, qui simule une variable aléatoire dont la loi est donnée par :

x	1	2	3	4	5
$\mathbf{P}(X=x)$	0.2	0.15	0.15	0.4	0.1

2. Écrire une généralisation de cette fonction, `X(T)`, qui simule une variable aléatoire dont la loi est définie par le tableau `T` de deux lignes. La première ligne est une suite strictement croissante de nombres (les (x_i)) ; la second ligne est une suite de nombres positifs dont la somme vaut 1 (les (p_i)).

3. Donner le script d'une fonction `empirique(S)` qui détermine la loi empirique d'une v.a.d.f. dont on connaît des simulations.
Valider les simulations des questions 1. et 2..

Exercice 4 Soient n et b deux entiers avec $n \geq 1$ et $b \geq 2$. On considère une urne contenant n boules noires et b boules blanches, toutes indiscernables. Un joueur A effectue des tirages successifs d'une boule sans remise dans l'urne jusqu'à obtenir une boule blanche. Il laisse alors la place au joueur B qui effectue des tirages successifs d'une boule avec remise dans l'urne jusqu'à obtenir une boule blanche.
On note X la variable aléatoire réelle égale au nombre de boules noires tirées par A avant de tirer une boule blanche et on appelle Y la variable aléatoire réelle égale au nombre de boules noires tirées par B avant de tirer une boule blanche (s'il ne reste plus de boule noire, on a donc $Y = 0$).
1. Écrire le script d'une fonction `tirage(n,b)` qui simule X.
2. En déduire, le script d'une fonction `partie(n,b)` qui retourne les valeurs de X et Y.
3. Représenter les lois empiriques des variables aléatoires X et Y. On utilisera l'instruction `bar`.



Exercice 5 Écrire une fonction `FR(X,P)` qui affiche la fonction de répartition associé à la variable donne les valeurs sont listées (par ordre croissant) dans `X` et dont la loi associée est donnée par la liste `P`.
Vous avez la liberté de prendrez toute initiative afin de rendre le graphique convenable.

Exercice 6 *Détermination des lois "usuelles"*
1. La loi binomiale $\mathscr{B}(n,p)$: nombre de succès dans un tirage avec remise.
Nous construisons par récurrence les vecteurs $(L_k)_{0\leq k\leq n} \in \left(\mathbb{R}^{n+1}\right)^{n+1}$ où $L_k = (x_{k,j})_{0\leq k\leq n}$ qui représente la loi de $X_k \hookrightarrow \mathscr{B}(k,p)$ sur le support $[\![0,n]\!]$. Ainsi,

$$\forall k,j \in [\![0,n]\!], \qquad x_{k,j} = \mathbf{P}(X_k = j)$$

$x_{k,j}$ est la probabilité d'obtenir j succès lors de k tirages avec remise dans une urne contenant une proportion p de boules gagnantes.
- initialisation : $L_0 = (1,0,\cdots,0) \in \mathbb{R}^{n+1}$
- pour $k = 1$, cela revient à la loi de Bernoulli, compléter :

$$L_1 = (\,?\,,\,?\,,0,\ldots,0)$$

- pour $k,j \in [\![1,n]\!]$, compléter les relations :

$$x_{k,0} = \ldots x_{k-1,0} \quad \text{et} \quad x_{k,l} = \ldots x_{k-1,l-1} + \ldots x_{k-1,l}$$

En déduire une fonction `loi_bin(n,p)` qui retourne la liste L_n.

2. La loi hypergéométrique $\mathcal{H}(N,n,p)$ (avec $pN \in \mathbb{N}^*$) : nombre de succès dans un tirage sans remise.
Avec les mêmes notations, $x_{k,j}$ est la probabilité d'obtenir j succès lors de k tirages sans remise dans une urne contenant une proportion p de boules gagnantes et N boules en tout. On trouve :

- initialisation : $L_0 = (1,0,\cdots,0) \in \mathbb{R}^{n+1}$
- pour $k=1$, cela revient à la loi de Bernoulli, compléter :

$$L_1 = (\ ?\ ,\ ?\ ,0,\ldots,0)$$

- pour $k,j \in [\![1,n]\!]$, compléter les relations :

$$x_{k,0} = \ldots x_{k-1,0} \quad \text{et} \quad x_{k,l} = \ldots x_{k-1,l-1} + \ldots x_{k-1,l}$$

En déduire une fonction `loi_hypergeo(N,n,p)` qui retourne la liste L_n.
3. La loi de premier succès dans un tirage sans remise $\mathcal{PS}(N,p)$ (avec $pN \in \mathbb{N}^*$).
Avec les mêmes notations mais sur des vecteurs de taille $N+1$, $x_{k,j}$ est la probabilité d'obtenir le première succès au j-ème triage lors de k tirages sans remise dans une urne contenant une proportion p de boules gagnantes et N boules au total. Le rang 0 correspond à la probabilité de ne pas obtenir de boule gagnante. On trouve :

- initialisation : $L_0 = (1,0,\cdots,0) \in \mathbb{R}^{N+1}$
- pour $k=1$, cela revient à la loi de Bernoulli, compléter :

$$L_1 = (\ ?\ ,\ ?\ ,0,\ldots,0)$$

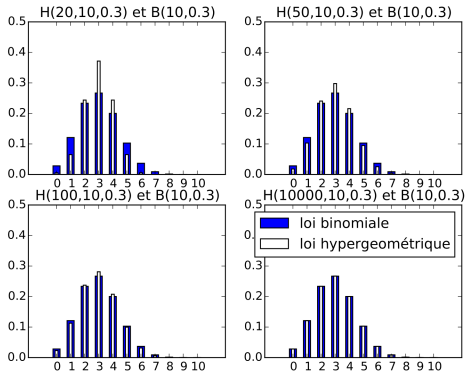
- pour $k,j \in [\![1,n]\!]$, compléter les relations :

$$\forall j \in [\![0,N]\!] \setminus \{0,k\}, \qquad x_{k,j} = \ldots$$

$$x_{k,0} = \ldots x_{k-1,0} \quad \text{et} \quad x_{k,k} = \ldots x_{k-1,0}$$

En déduire une fonction `loi_PS(N,n,p)` qui retourne la liste L_n.

Exercice 7 Illustrer la convergence en loi de la loi hypergéométrique $\mathcal{H}(N,n,p)$ vers la loi binomiale $\mathcal{B}(n,p)$ lorsque $N \to +\infty$ (cf exo de TD).
En particulier, on pourra tracer avec les diagrammes en bâtons les lois $\mathcal{B}(n,p)$ et $\mathcal{H}(N,n,p)$ telles que $n=10$, $p=0,3$ et $N \in \{20,50,100,10000\}$.



Autres simulations

Exercice 8 Écrire une fonction simulant chacun des objets suivant :
1. une permutation aléatoire sur $[\![0,n-1]\!]$: `permutation(n)`
2. un tirage de l'euro-million, `euroM()` : 5 numéros sur $[\![1,50]\!]$ et deux étoiles sur $[\![1,11]\!]$

Exercice 9
Écrire le script d'une fonction, `tirage(n)`, simulant le tirage de deux boules (sans remise) dans une urne contenant n boules numérotées de 1 à n.

➢ Comment faire la différence entre un tirage avec ordre ou sans ordre ?

Exercice 10 Soit $n \in \mathbb{N}^*$. Une urne $\mathcal{U}_n$ contient n boules numérotées de 1 à n. On effectue dans cette urne une succession de tirages d'une boule, en appliquant la règle suivante : si une boule tirée porte le numéro k, avant de procéder au tirage suivant, on enlève toutes les boules dont le numéro est supérieur ou égal à k.

On note X_n la variable aléatoire égale au nombre de tirages nécessaires pour vider l'urne $\mathcal{U}_n$ de toutes ses boules.

Compléter le programme suivant afin qu'il simule l'expérience :

```
def X(n):
    x=...
    numero=rd.randint(...)
    while ...:
        x=...
        numero=...
    return(x)
```

Exercice 11 Considérons l'expérience suivante : on lance n fois une pièce donnant *Pile* avec probabilité p_1 et on note X le nombre de fois où l'on obtient *Pile*.

Puis, on lance X fois une seconde pièce donnant *Pile* avec probabilité p_2 et on note Y le nombre de fois où l'on obtient *Pile* lors de cette seconde série de lancers.
1. Compléter cette fonction qui simule Y :

```
def Y(n,p1,p2)
    X=0
    for i in range(n):
        if rd.rand()<p1:
            X=...
    Y=0
    for i in ...:
        if ...:
            ...
    return(Y)
```

2. Déterminer le nombre moyen d'appels de rand effectués dans cette fonction.
3. Utilisant l'instruction `binomial`, donner une autre fonction pour simuler Y.

Exercice 12 On désigne par n un entier naturel supérieur ou égal à 2. On dispose de n urnes, numérotées de 1 à n, contenant chacune n boules. On répète n épreuves, chacune consistant à choisir une urne au hasard et à en extraire une

boule au hasard. On suppose que les choix des urnes sont indépendants les uns des autres.
Pour tout $i \in [\![1,n]\!]$,
- on note X_i la variable aléatoire prenant la valeur 1 si l'urne numérotée i contient toujours n boules au bout de ces n épreuves, et qui prend la valeur 0 sinon ;
- et on note N_i le nombre de boules manquantes dans l'urne i à l'issue de l'expérience.

Proposer une fonction urnes(n) qui simule l'expérience et affiche les valeurs prises par X_1 et N_1 pour le paramètre n.

Exercice 13 *Schéma de Markov*
Deux personnes sont perdues dans un labyrinthe composé de cinq pièces (P_i pour $i \in [\![0,4]\!]$) réparties en cercle. Deux pièces voisines sont reliées par un couloir. A l'instant $n = 0$, les deux personnes sont dans les pièces voisines P_0 et P_1. Elles partent à la recherche l'une de l'autre selon les règles suivantes :
- une personne à peu aller dans chacune des pièces voisines avec un probabilité de $\frac{1}{2}$;
- les déplacement des deux personnes sont simultanés, indépendants jusqu'à leur éventuelle rencontre. Ensuite, elles se déplace ensemble ;
- les personnes ne peuvent pas se voir dans les couloirs.

1. Écrire un script dep(n) retournant deux listes contenant les n premiers déplacements des deux personnes.
2. Représenter sur un graphiques plusieurs simulations des ces déplacements.

Exercice 14 On considère une expérience consistant à lancer une pièce n fois de manière consécutive. L'expérience en question est décrite par le programme PYTHON suivant pour la valeur $n = 10$ (et répétée ici 1000 fois).

```
Nexp=1000
n=10
u=np.floor(2*rd.rand(n,Nexp))+1
test=0
for k in range(Nexp):
    ok=1
    for i in range(1,n):
        if (u[i-1,k]==1) and (u[i,k]==1):
            ok=0
    test=test+ok
Pn=test/Nexp
print(Pn)
```

1. Que représente la variable Pn affichée par le programme ?
2. En notant P_n le résultat de l'expérience précédent pour une valeur de n quelconque, montrer que

$$P_n = \frac{1}{2}P_{n-1} + \frac{1}{4}P_{n-2}$$

puis trouver la limite de P_n quand n tend vers l'infini.
3. Adapter la ligne 3 à l'étude d'un lancer d'un dé, en précisant la caractéristique étudiée.

Exercice 15 *Rang d'une première paire dans un tirage avec remise* On considère une urne contenant une boule blanche et trois boules noires. On effectue des tirages successifs d'une boule avec remise.

1. Ecrire un programme simulant ces tirages et déterminant le rang où pour la première fois on obtient une deuxième boule blanche consécutive.
2. Identifier la loi théorique.
3. Définir la fonction compare(n) qui donne sur le même graphique, la représentation de la loi théorique et de celle empirique obtenue après n simulations.

Exercice 16 *Simulation d'un jeu* Une urne contient b boules bleue, v boules vertes et r boules rouges. Le jeu consiste à effectuer une succession de tirages d'une boule dans l'urne, sans remise, jusqu'à ce que la boule tirée soit :
- ou bleue, auquel cas la partie est gagnée,
- ou verte, et la partie est perdue.

La longueur de la partie est le nombre de boules sorties à la fin de la partie.
Compléter le programme PYTHON qui simule le déroulement d'une partie, retournant sa longueur et son échéance :

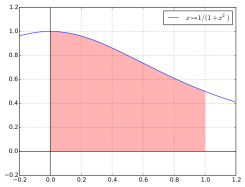
```
def urneVBR(b,v,r):
    ...
    x='R'
    while x==...:
        y=y+1
        alea=rd.randint(1,...)
        if alea<=b:
            x='B'
        elif ...
            x='V'
        else:
            ...
    if x=='V':
        return(y,'PERDU')
    else:
        return(...)
```

Méthode Monté-Carlo

Méthode : Considérant $X \hookrightarrow \mathcal{U}([a,b])$, (X_k) des copies de X et une fonction g continue sur $[a,b]$. Alors

$$\frac{1}{n}\sum_{k=1}^{n} g(X_k) \underset{n\to+\infty}{\longrightarrow} \mathbf{E}(g(X)) = \frac{1}{b-a}\int_a^b g(t)\mathrm{d}t$$

La loi faible de grands nombres donne que *la moyenne empirique est une estimateur de l'espérance*.



Exercice 17 *Calcul d'une intégrale*

1. Écrire le script d'une fonction `Monte_Carlo(f,a,b,n)` qui retourne une approximation de l'intégrale de g sur $[a,b]$ par la méthode de Monté-Carlo, en prenant `n` simulations par unité de mesure.
2. Estimer la vitesse de convergence sous la forme de $\mathcal{O}(n^{-\alpha})$.

Exercice 18 **VIDEO** On lance n fléchettes sur une cible carrée de côté unitaire. On dénombre celles qui ont atteints le quart de disque centré en un sommet et de rayon 1 : X_n

1. Quelle est la limite de la suite $\left(\frac{X_n}{n}\right)$?
2. Écrire le script d'une fonction `flechettes(n)` qui simule l'expérience et retourne $\frac{4X_n}{n}$.