Feuille de TP n° 1

Variables aléatoires à densité avec Python (corrections)

I Représentation graphique des lois à densité usuelles

Exercice 1 (Représentation de lois uniformes).

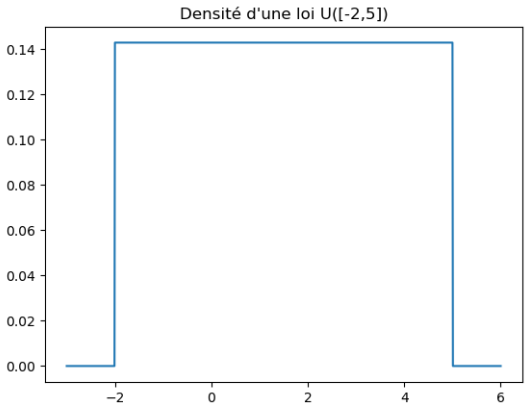
1) Écrire une fonction qui prend en argument deux réels a et b tels que $a < b$ et qui trace la densité d'une variable aléatoire de loi $\mathcal{U}([a\,;b])$ sur l'intervalle $[a-1\,;b+1]$.
2) Écrire une fonction qui prend en argument deux réels a et b tels que $a < b$ et qui trace la fonction de répartition d'une variable aléatoire de loi $\mathcal{U}([a\,;b])$ sur l'intervalle $[a-1\,;b+1]$.
3) Tester les fonctions précédentes pour quelques valeurs de a et b.

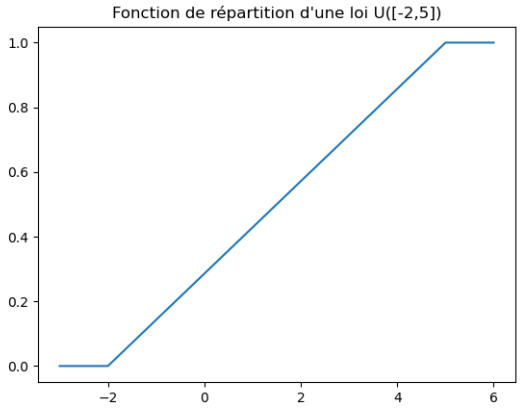
Correction :

1)
```
import numpy as np
import matplotlib.pyplot as plt
def DensiteUnif(a,b):
    def f(x):
        if x<a or x>b:
            return 0
        else:
            return 1/(b-a)
    X=np.linspace(a-1,b+1,1000)
    Y=[f(x) for x in X]
    plt.plot(X,Y)
    plt.title("Densité d'une loi U(["+str(a)+","+str(b)+"])")
    plt.show()
```

2)
```
import numpy as np
import matplotlib.pyplot as plt
def FRepUnif(a,b):
    def F(x):
        if x<a:
            return 0
        elif x>b:
            return 1
        else:
            return (x-a)/(b-a)
    X=np.linspace(a-1,b+1,1000)
    Y=[F(x) for x in X]
    plt.plot(X,Y)
    plt.title("Fonction de répartition d'une loi U(["+str(a)+","+str(b)+"])")
    plt.show()
```

3)





Exercice 2 (Représentation de lois exponentielles).

1) Soient $a > 0$ et X une variable aléatoire de loi $\mathcal{E}(a)$. Déterminer $t_a \in \mathbb{R}$ (le plus petit possible) tel que $\mathbb{P}(X > t_a) \leqslant 10^{-3}$.
2) Écrire une fonction qui prend en argument $a > 0$ et qui trace la densité d'une variable aléatoire de loi $\mathcal{E}(a)$ sur l'intervalle $[-1\,;t_a]$.
3) Écrire une fonction qui prend en argument $a > 0$ et qui trace la fonction de répartition d'une variable aléatoire de loi $\mathcal{E}(a)$ sur l'intervalle $[-1\,;t_a]$.
4) Tester les fonctions précédentes pour quelques valeurs de a.

Correction :

1) Pour tous $a > 0$ et $t > 0$, on a $\mathbb{P}(X > t_a) = 1 - F_X(t) = e^{-at}$. Ainsi $\mathbb{P}(X > t_a) \leqslant 10^{-3}$ si et seulement si $-at \leqslant -3\ln(10)$ si et seulement si $t \geqslant \dfrac{3}{a}\ln(10)$.

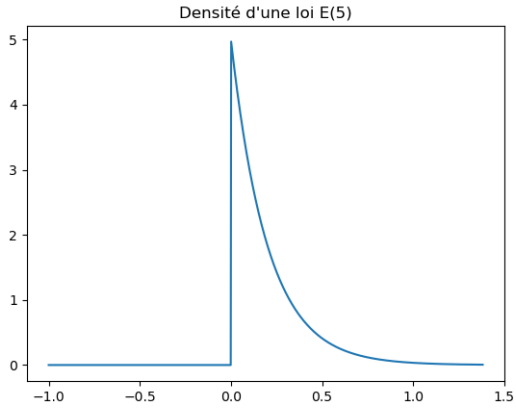
2)

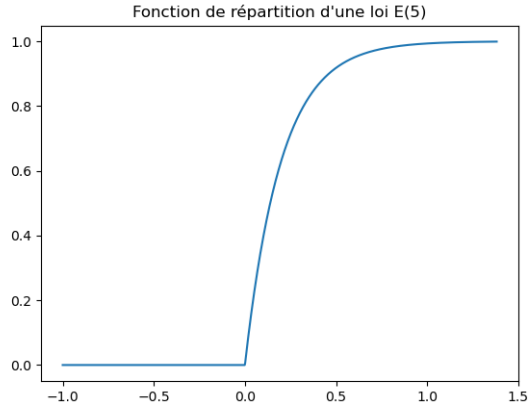
```
import numpy as np
import matplotlib.pyplot as plt
def DensiteExpo(a):
    def f(x):
        if x<0:
            return 0
        else:
            return a*np.exp(-a*x)
    ta=3*np.log(10)/a
    X=np.linspace(-1,ta,1000)
    Y=[f(x) for x in X]
    plt.plot(X,Y)
    plt.title("Densité d'une loi E("+str(a)+")")
    plt.show()
```

3)

```
import numpy as np
import matplotlib.pyplot as plt
def FRepExpo(a):
    def F(x):
        if x<0:
            return 0
        else:
            return 1-np.exp(-a*x)
    ta=3*np.log(10)/a
    X=np.linspace(-1,ta,1000)
    Y=[F(x) for x in X]
    plt.plot(X,Y)
    plt.title("Fonction de répartition d'une loi E("+str(a)+")")
    plt.show()
```

4)





Exercice 3 (Représentation de lois normales).

1) Écrire une fonction qui prend en argument $m \in \mathbb{R}$ et $\sigma^2 > 0$ et qui trace la densité d'une variable aléatoire de loi $\mathcal{N}(m, \sigma^2)$ sur l'intervalle $[m - 4\sigma\,; m + 4\sigma]$.

2) a) Exprimer la fonction de répartition d'une variable aléatoire de loi $\mathcal{N}(m, \sigma^2)$ en fonction de Φ.

   b) Écrire une fonction qui prend en argument $m \in \mathbb{R}$ et $\sigma^2 > 0$ et qui trace la fonction de répartition d'une variable aléatoire de loi $\mathcal{N}(m, \sigma^2)$ sur l'intervalle $[m - 4\sigma\,; m + 4\sigma]$.

   On utilisera la fonction `Phi` *implémentée dans le TP précédent ou bien la fonction* `ndtr` *de la bibliothèque* `scipy.special` *(importée sous le nom* `sp` *par exemple).*

3) Tester les fonctions précédentes pour quelques valeurs de m et σ^2.

Correction :

1)

```
import numpy as np
import matplotlib.pyplot as plt
def DensiteNormale(m,s2):
    def f(x):
        return np.exp(-(x-m)**2/(2*s2))/np.sqrt(2*np.pi*s2)
    X=np.linspace(m-4*s2**(1/2),m+4*s2**(1/2),1000)
    Y=[f(x) for x in X]
    plt.plot(X,Y)
    plt.title("Densité d'une loi N("+str(m)+","+str(s2)+")")
    plt.show()
```

2) a) Si X suit une loi $\mathcal{N}(m, \sigma^2)$, alors $\dfrac{X - m}{\sigma}$ suit une loi $\mathcal{N}(0, 1)$. Ainsi, pour tout $x \in \mathbb{R}$,

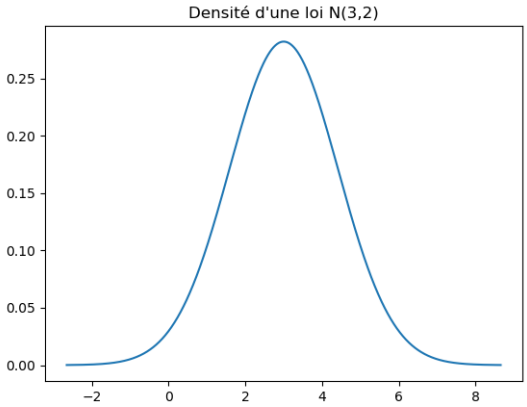
$$F_X(x) = \mathbb{P}(X \leqslant x) = \mathbb{P}\left(\frac{X - m}{\sigma} \leqslant \frac{x - m}{\sigma}\right) = \Phi\left(\frac{x - m}{\sigma}\right).$$

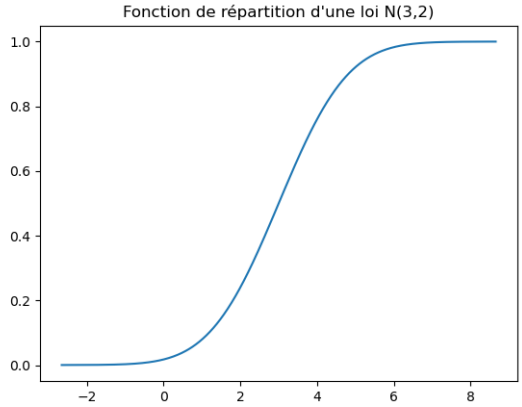
b)

```
import numpy as np
import matplotlib.pyplot as plt
def Phi(x):
    def phi(t):
        return np.exp(-t**2/2)/np.sqrt(2*np.pi)
    n=int(x**2/(2*0.001*np.sqrt(2*np.pi*np.e)))+1
    S=0
    for k in range(1,n+1):
        S=S+phi(x*k/n)
    S=1/2+x*S/n
    return S

def FRepNormale(m,s2):
    s1=s2**(1/2)
    def F(x):
        return Phi((x-m)/s1)
    X=np.linspace(m-4*s1,m+4*s1,1000)
```

```
    Y=[F(x) for x in X]
    plt.title("Fonction de répartition d'une loi N("+str(m)+","+str(s2)+")")
    plt.plot(X,Y)
    plt.show()
```

3)





Exercice 4 (Représentation de lois gamma).

1) Écrire une fonction qui prend en argument un **entier naturel** n non nul et qui trace la densité d'une variable aléatoire de loi $\gamma(n)$ sur l'intervalle $[-1\,;5n]$.
2) Écrire une fonction qui prend en argument un **entier naturel** n supérieure ou égale à 2 (pour $n=1$, voir cf. exercice 2) et qui trace la fonction de répartition d'une variable aléatoire de loi $\gamma(n)$ sur l'intervalle $[-1\,;5n]$.
   On utilisera la méthode des rectangles.
3) Tester les fonctions précédentes pour quelques valeurs de n.

Correction :

1) Pour tout $n\in\mathbb{N}^*$, $\Gamma(n)=(n-1)!$.

```
import numpy as np
import matplotlib.pyplot as plt
def DensiteGamma(n):
    Facto=np.prod([k for k in range(1,n)])
    def f(x):
        if x<0:
            return 0
        else:
            return x**(n-1)*np.exp(-x)/Facto
    X=np.linspace(-1,5*n,1000)
    Y=[f(x) for x in X]
    plt.plot(X,Y)
    plt.title("Densité d'une loi Gamma("+str(n)+")")
    plt.show()
```

2) Soit $n\in\mathbb{N}^*$. Notons f_n la densité d'une loi Gamma de paramètre n. Elle est deux fois dérivable sur $\mathbb{R}_+^*$. Pour tout $x\in\mathbb{R}_+^*$,

$$f_n'(x)=\frac{(n-1)x^{n-2}e^{-x}-x^{n-1}e^{-x}}{(n-1)!}=\frac{((n-1)-x)x^{n-2}e^{-x}}{(n-1)!}.$$

Ensuite

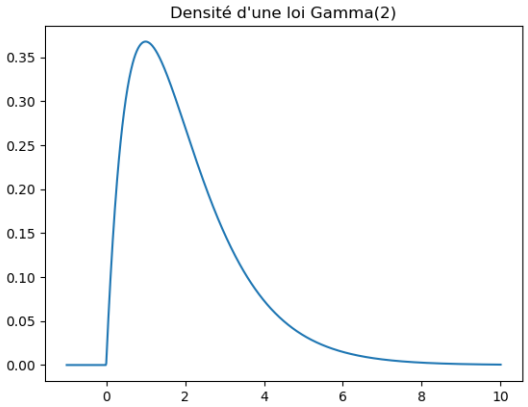
$$\begin{aligned}f_n''(x)&=\frac{(n-1)(n-2)x^{n-3}e^{-x}-(n-1)x^{n-1}-((n-1)-x)x^{n-2}e^{-x}}{(n-1)!}\\&=\frac{((n-1)(n-2)-2(n-1)x+x^2)x^{n-3}e^{-x}}{(n-1)!}.\end{aligned}$$

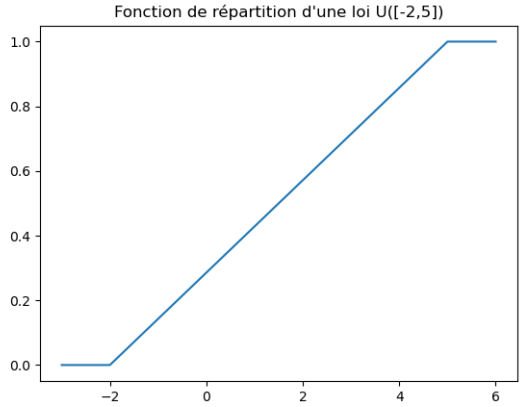
Le trinôme $X^2-2(n-1)X+(n-1)(n-2)$ admet pour discriminant $4(n-1)^2-4(n-1)(n-2)=4(n-1)\geqslant 0$. Il admet donc deux racines (identiques si $n=1$) : $\dfrac{2(n-1)\pm\sqrt{4(n-1)}}{2}=n-1+\sqrt{n-1}$. Ainsi f_n' est croissante sur $\left]0\,;n-1-\sqrt{n-1}\right]$, décroissante sur $\left[n-1-\sqrt{n-1}\,;n-1+\sqrt{n-1}\right]$ et croissante sur $\left[n-1+\sqrt{n-1}\,;+\infty\right[$. Puisque sa limite est nulle en $+\infty$ (par croissantes comparées si $n>2$), le maximum de f_n' est

$$\max\{f_n'(n-1-\sqrt{n-1}),|f_n'(n-1+\sqrt{n-1})|\}.$$

```python
import numpy as np
import matplotlib.pyplot as plt

#On commencer par le max de gn'=fn'*(n-1)! :
import numpy as np
def max_gnp(n):
    def gnp(x):
        return (n-1-x)**(n-2)*np.exp(-x)
    x1=n-1+np.sqrt(n-1)
    x2=n-1-np.sqrt(n-1)
    m=max(gnp(x1),np.abs(gnp(x2)))
    return m

#Puis la méthode des rectangles pour Gn=Fn*(n-1)! (attention n est déjà prise) :
def G(n,x):
    if x<=0:
        return 0
    else:
        N=int(x**2/(2*0.001)*max_gnp(n))+1
        S=0
        for k in range(1,N+1):
            S=S+(x*k/N)**(n-1)*np.exp(-x*k/N)
    return S*x/N

def FRepGamma(n):
    X=np.linspace(-1,5*n,1000)
    Facto=np.prod([k for k in range(1,n)])
    Y=[G(n,x)/Facto for x in X]
    plt.plot(X,Y)
    plt.title("Fonction de répartition d'une loi Gamma("+str(n)+")")
    plt.show()
```

3)





II Simulation de variables aléatoires à densité

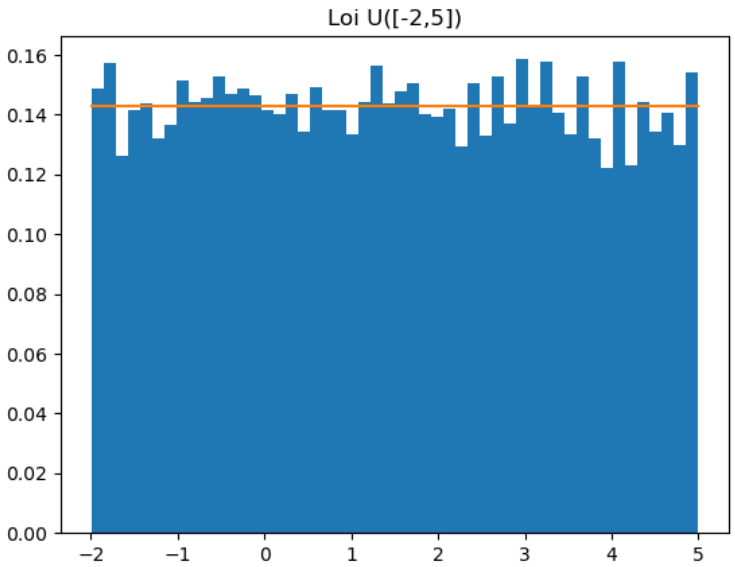
1) Utilisation de `numpy.random`

Exercice 5 (« Vérification » des simulations).

1) Écrire une fonction qui prend en argument des réels a et b (avec $a < b$), qui simule un échantillon de 10000 variables aléatoires de loi $\mathcal{U}([a\,;b])$ en utilisant `rd.random`, qui trace l'histogramme empirique renormalisé de cet échantillon et qui lui superpose le graphe d'une densité d'une variable de loi $\mathcal{U}([a\,;b])$. Tester avec plusieurs valeurs de a et b.
2) Écrire une fonction qui prend en argument un réel strictement positif a, qui simule un échantillon de 10000 variables aléatoires de loi $\mathcal{E}(a)$ en utilisant `rd.exponential`, qui trace l'histogramme empirique renormalisé de cet échantillon et qui lui superpose le graphe d'une densité d'une variable de loi $\mathcal{E}(a)$. Tester avec plusieurs valeurs de a.
3) Écrire une fonction qui prend en argument deux réels m et σ^2 avec $\sigma^2 > 0$, qui simule un échantillon de 10000 variables aléatoires de loi $\mathcal{N}(m, \sigma^2)$ en utilisant `rd.normal`, qui trace l'histogramme empirique renormalisé de cet échantillon et qui lui superpose le graphe d'une densité d'une variable de loi $\mathcal{N}(m, \sigma^2)$. Tester avec plusieurs valeurs de m et σ^2.
4) Écrire une fonction qui prend en argument un entier naturel strictement positif n, qui simule un échantillon de 10000 variables aléatoires de loi $\gamma(n)$ en utilisant `rd.gamma`, qui trace l'histogramme empirique renormalisé de cet échantillon et qui lui superpose le graphe d'une densité d'une variable de loi $\gamma(n)$. Tester avec plusieurs valeurs de n.

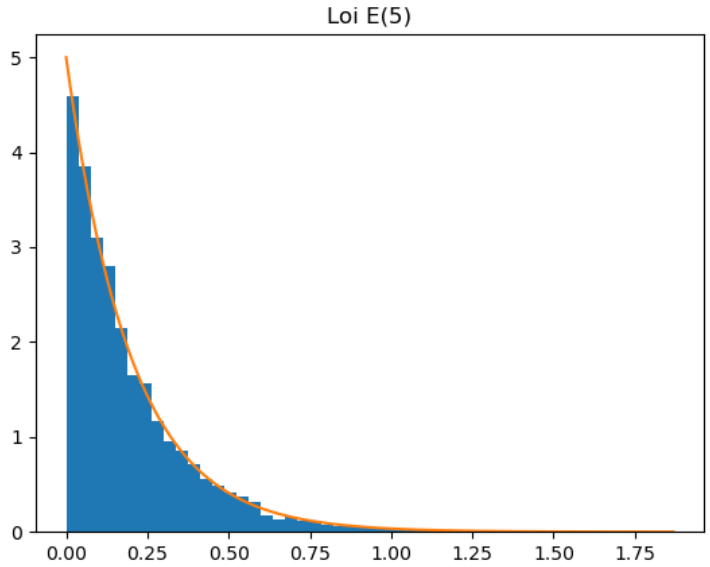
Correction :

1)
```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
def VerifUnif(a,b):
    S=(b-a)*rd.random(10000)+a
    plt.hist(S,50,density=True)
    X=[np.min(S),np.max(S)]
    Y=[1/(b-a),1/(b-a)]
    plt.plot(X,Y)
    plt.title("Loi U(["+str(a)+","+str(b)+"])")
    plt.show()
```



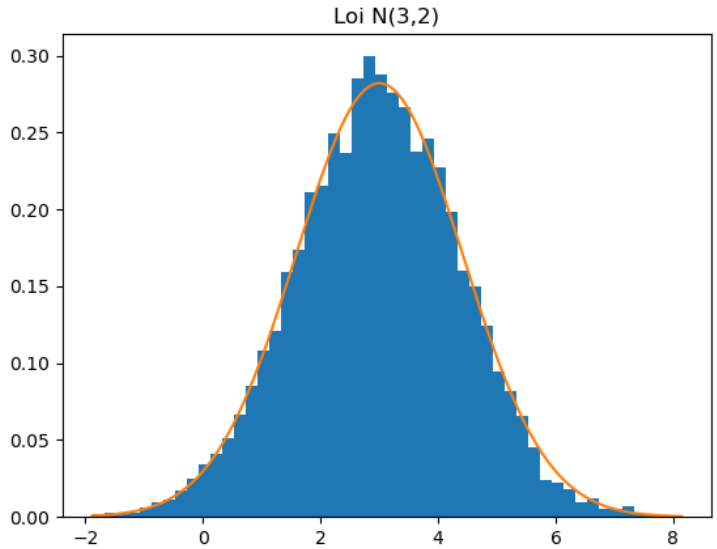
2)
```
import numpy as np
import numpy.random as rd
```

```
import matplotlib.pyplot as plt
def VerifExpo(a):
    #Attention : l'argument est l'inverse du paramètre
    S=rd.exponential(1/a,10000)
    plt.hist(S,50,density=True)
    X=np.linspace(np.min(S),np.max(S),1000)
    Y=[a*np.exp(-a*x) for x in X]
    plt.plot(X,Y)
    plt.title("Loi E("+str(a)+")")
    plt.show()
```



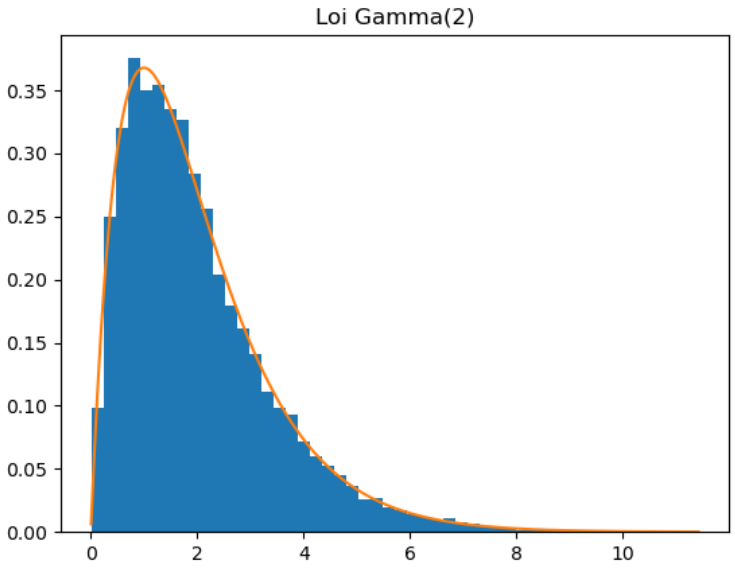
3)

```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
def VerifNormale(m,s2):
    #Attention : les arguments sont la moyenne et l'écart type
    S=rd.normal(m,np.sqrt(s2),10000)
    plt.hist(S,50,density=True)
    X=np.linspace(np.min(S),np.max(S),1000)
    Y=[np.exp(-(x-m)**2/(2*s2))/np.sqrt(2*np.pi*s2) for x in X]
    plt.plot(X,Y)
    plt.title("Loi N("+str(m)+","+str(s2)+")")
    plt.show()
```



4)

```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
def VerifGamma(n):
    #Attention : il y a un argument d'échelle qui vaut 1
    S=rd.gamma(n,1,10000)
    plt.hist(S,50,density=True)
    X=np.linspace(np.min(S),np.max(S),1000)
    facto=np.prod([k for k in range(1,n)])
    Y=[x**(n-1)*np.exp(-x)/facto for x in X]
    plt.plot(X,Y)
    plt.title("Loi Gamma("+str(n)+")")
    plt.show()
```



Exercice 6 (Somme de loi exponentielles).

1) Écrire une fonction en Python qui prend en argument un entier naturel n non nul et qui renvoie la somme de n variables aléatoire de lois $\mathcal{E}(1)$.
2) Pour différentes valeurs de n, exécuter 10000 fois cette fonction, construire l'histogramme empirique renormalisé de ces réalisations et lui superposer le graphe d'une densité d'une variable aléatoire de loi $\gamma(n)$.

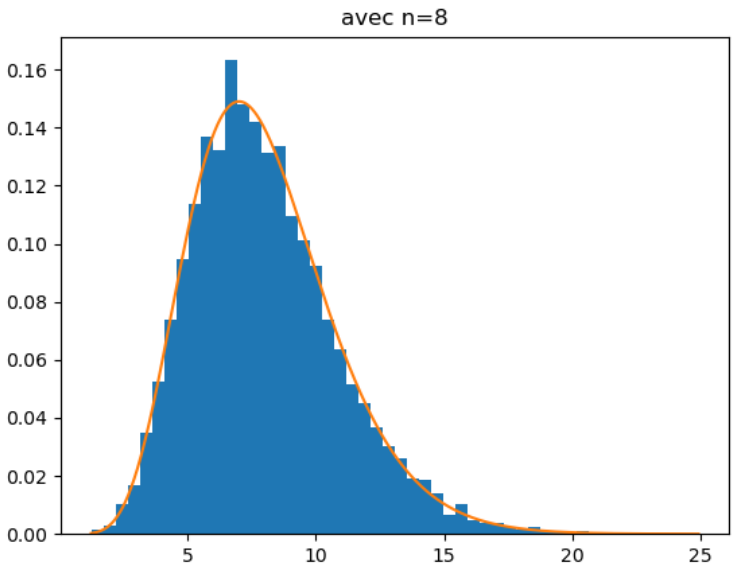
Correction :

1)

```
import numpy as np
import numpy.random as rd
def SommeExp(n):
    return np.sum(rd.exponential(1,n))
```

2)

```
import matplotlib.pyplot as plt
def TraceSommeExp(n):
    L=[]
    for k in range(10000):
        L.append(SommeExp(n))
    plt.hist(L,50,density=True)
    X=np.linspace(np.min(L),np.max(L),1000)
    Facto=np.prod([k for k in range(1,n)])
    Y=[x**(n-1)*np.exp(-x)/Facto for x in X]
    plt.plot(X,Y)
    plt.title("avec n="+str(n))
    plt.show()
```



On constate que la densité d'une loi gamma de paramètre n se superpose bien à l'histogramme. Cela illustre le fait qu'une somme de n variables aléatoires de loi exponentielle de paramètre 1 suit une loi gamma de paramètre n.

Exercice 7. On dispose d'un bâton d'un mètre. On le casse en deux morceaux en choisissant le point de cassure au hasard (uniformément). On note X la longueur du plus petit morceau et Y la longueur du plus grand morceau.

1) a) Écrire une fonction qui prend en argument $N \in \mathbb{N}^*$ et qui renvoie un vecteur (une matrice ligne et non pas une liste) contenant N réalisations indépendantes de X.

   b) Tracer l'histogramme empirique renormalisé de 10000 réalisations de X. Que peut-on conjecturer ?

   c) Recopier et exécuter le script suivant :

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  N=10000
4  X=SimulX(N)
5  T=np.linspace(0,1,1000)
6  F=[]
7  for t in T:
8      F.append(np.sum(X<=t)/N)
9  plt.plot(T,F)
10 plt.show()
```

   Que fait ce script (en particulier que font les lignes 7 et 8) ? Que peut-on conjecturer de nouveau ?

   d) Prouver la conjecture.

2) a) Tracer l'histogramme empirique renormalisé de 10000 réalisations de $Z = X/Y$.

   b) Déterminer la fonction de répartition de Z puis une densité de Z.

   c) Superposer le graphe de cette densité à l'histogramme de la question a.

   d) Superposer le graphe fonction de répartition empirique (qu'est ce qu'on peut entendre par là ?) de cet échantillon et le graphe de la fonction de répartition de Z ? Qu'observe-t-on ?

Correction :

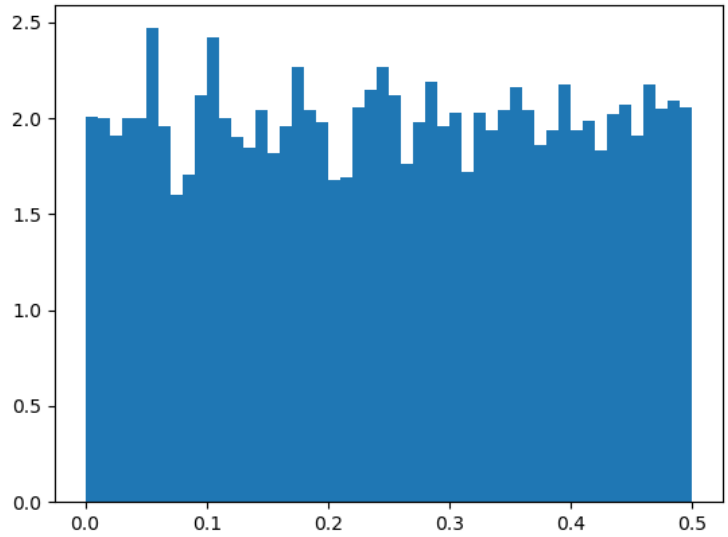
1) a)

```
1 import numpy as np
2 import numpy.random as rd
3 def SimulX(N):
4     L=[]
```

```
    for k in range(N):
        U=rd.random()
        L.append(min(U,1-U))
    return np.array(L)
```

b)

```
import matplotlib.pyplot as plt
N=10000
X=SimulX(N)
plt.hist(X,50,density=True)
plt.show()
```



On superposerait très bien une fonction constant égale à 2 sur l'intervalle $[0\,;1/2]$, et égale à 0 sinon. On conjecture ainsi que $X \hookrightarrow \mathcal{U}([0\,;1/2])$.

c) Ce script simule $N = 10000$ réalisations $X_1, \dots, X_N$ de la variable aléatoire X et les stocke dans la liste `X`. Ensuite, pour chaque réel `t` du vecteur `T` (qui contient 1000 réels régulièrement espacés entre 0 et 1), elle ajoute dans la liste `F` la proportion de réalisations de X qui sont inférieures ou égales à t. En effet la commande `X<=t` transforme la liste `X` en une liste de booléens où les coordonnées inférieurs à `t` deviennent `True` et les autres `False`. Ainsi `np.sum(X<=t)` compte le nombre de coordonnées de `X` qui sont inférieures ou égales à `t`. Ainsi le script représente graphiquement 1000 valeurs de la fonction

$$t \longmapsto \frac{\text{card}(\{i \in [\![1\,;N]\!] \,|\, X_i \leqslant t\})}{N}.$$

D'après la loi faible des grands nombres, cette proportion est une approximation de la probabilité que X soit inférieure ou égale à t, c'est-à-dire $F_X(t)$. En exécutant ce script, on constate que la courbe représentée se superpose avec la fonction de répartition d'une variable aléatoire de loi $\mathcal{U}([0\,;1/2])$. On peut donc conjecturer que $X \hookrightarrow \mathcal{U}([0\,;1/2])$.

d) Calculons la fonction de répartition de X. Soit $x \in \mathbb{R}$. On applique la formule des probabilités totales avec le système complet d'événements $([U > 1/2], [U \leqslant 1/2])$:

$$\begin{aligned}\mathbb{P}(X \leqslant x) &= \mathbb{P}([X \leqslant x] \cap [U > 1/2]) + \mathbb{P}([X \leqslant x] \cap [U \leqslant 1/2]) \\ &= \mathbb{P}([1 - U \leqslant x] \cap [U > 1/2]) + \mathbb{P}([U \leqslant x] \cap [U \leqslant 1/2]) \\ &= \mathbb{P}([U \geqslant 1 - x] \cap [U > 1/2]) + \mathbb{P}([U \leqslant x] \cap [U \leqslant 1/2]).\end{aligned}$$

Faisons plusieurs cas :

- Si $x \leqslant 0$, alors $\mathbb{P}(X \leqslant x) = 0 + 0 = 0$.
- Si $x \in]0\,;1/2]$, $1 - x > \dfrac{1}{2}$ donc

$$\mathbb{P}(X \leqslant x) = \mathbb{P}(U > 1 - x) + \mathbb{P}(U \leqslant x) = 1 - \mathbb{P}(U \leqslant 1 - x) + \mathbb{P}(U \leqslant x) = 1 - (1 - x) + x = 2x.$$

- Si $x \geqslant \frac{1}{2}$, $1-x \leqslant \frac{1}{2}$ donc

$$\mathbb{P}(X \leqslant x) = \mathbb{P}(U > 1/2) + \mathbb{P}(U \leqslant 1/2) = \frac{1}{2} + \frac{1}{2} = 1.$$

Ainsi F_X est la fonction de répartition d'une variable aléatoire de loi $\mathcal{U}([0\,;1/2])$. Cela prouve la conjecture.

2) a)

```
import numpy as np
import numpy.random as rd
def SimulZ(N):
    L=[]
    for k in range(N):
        U=rd.random()
        if U>1/2:
            L.append((1-U)/U)
        else:
            L.append(U/(1-U))
    return np.array(L)

import matplotlib.pyplot as plt
N=10000
Z=SimulZ(N)
plt.hist(Z,50,density=True)
plt.show()
```

Cf. question 2c pour l'histogramme.

b) Calculons la fonction de répartition de Z. Déjà, il est clair que $0 \leqslant X \leqslant Y \leqslant 1$ par définition. En particulier Z prend ses valeurs dans l'intervalle $[0\,;1]$ en particulier. Ainsi :

- Si $x \leqslant 0$, $F_Z(x) = 0$.
- Si $x \geqslant 1$, $F_Z(x) = 1$.

Soit $x \in]0\,;1[$. On applique la formule des probabilités totales avec le système complet d'événements $([U > 1/2], [U \leqslant 1/2])$:

$$\begin{aligned}
\mathbb{P}(Z \leqslant x) &= \mathbb{P}([Z \leqslant x] \cap [U > 1/2]) + \mathbb{P}([Z \leqslant x] \cap [U \leqslant 1/2]) \\
&= \mathbb{P}([\frac{1-U}{U} \leqslant x] \cap [U > 1/2]) + \mathbb{P}([\frac{U}{1-U} \leqslant x] \cap [U \leqslant 1/2]) \\
&= \mathbb{P}([\frac{1}{U} \leqslant x+1] \cap [U > 1/2]) + \mathbb{P}([\frac{1-U}{U} \geqslant \frac{1}{x}] \cap [U \leqslant 1/2]) \\
&= \mathbb{P}([U \geqslant \frac{1}{1+x}] \cap [U > 1/2]) + \mathbb{P}([\frac{1}{U} \geqslant \frac{1}{x}+1] \cap [U \leqslant 1/2]) \\
&= \mathbb{P}([U \geqslant \frac{1}{1+x}] \cap [U > 1/2]) + \mathbb{P}([U \leqslant \frac{1}{1+1/x}] \cap [U \leqslant 1/2])
\end{aligned}$$

Lorsque x parcours $]0\,;1[$, $\frac{1}{2} \leqslant \frac{1}{1+x} \leqslant 1$ et $0 < \frac{1}{1+1/x} \leqslant \frac{1}{2}$. Ainsi :

$$\mathbb{P}(Z \leqslant x) = \mathbb{P}(U \geqslant \frac{1}{1+x}) + \mathbb{P}(U \leqslant \frac{1}{1+1/x}) = 1 - \frac{1}{1+x} + \frac{1}{1+1/x} = \frac{2x}{1+x}.$$

Résumons :

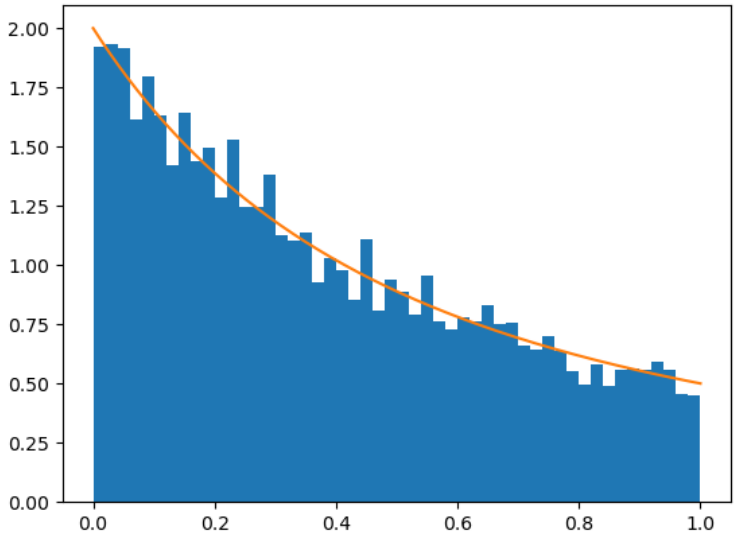
$$\forall x \in \mathbb{R}, \qquad F_Z(x) = \begin{cases} 0 & \text{si} \quad x \leqslant 0 \\ \dfrac{2x}{1+x} & \text{si} \quad x \in]0\,;1[\\ 0 & \text{si} \quad x \geqslant 1 \end{cases}$$

La fonction F_Z est de classe C^1 sur $\mathbb{R}$ sauf éventuellement en 0 et 1. Puisque $\lim\limits_{x \to 0^+} \dfrac{2x}{1+x} = 0$ et $\lim\limits_{x \to 1^-} \dfrac{2x}{1+x} = 1$, on en déduit que F_Z est continue sur $\mathbb{R}$ tout entier. Ainsi c'est une variable à densité. En dérivant, on trouve qu'une densité de Z est :

$$f_Z : x \longmapsto \begin{cases} \dfrac{2}{(1+x)^2} & \text{si} \quad x \in]0\,;1[\\ 0 & \text{sinon} \end{cases}$$

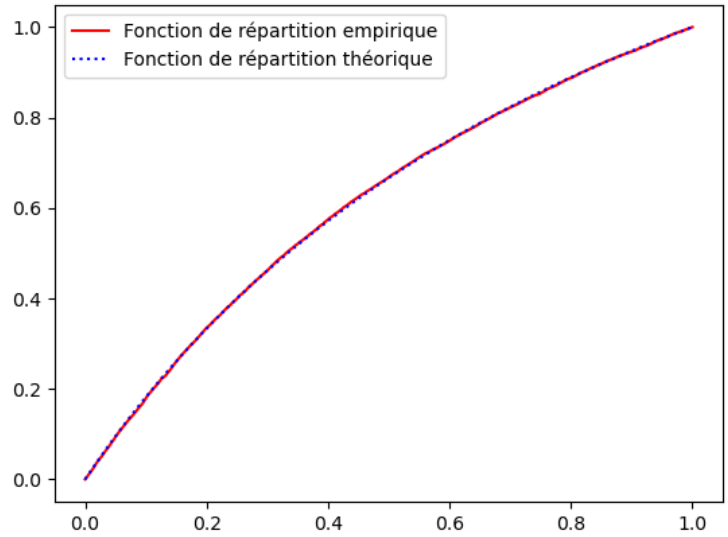
c)

```
N=10000
Z=SimulZ(N)
plt.hist(Z,50,density=True)
Abs=np.linspace(np.min(Z),np.max(Z),1000)
Ord=[2/(1+x)**2 for x in Abs]
plt.plot(Abs,Ord)
plt.show()
```



d) On s'inspire de la méthode de la question 1c :

```
#Fonction de répartition empirique
N=10000
Z=SimulZ(N)
T=np.linspace(0,1,1000)
F=[]
for t in T:
    F.append(np.sum(Z<=t)/N)
plt.plot(T,F,'r',label="Fonction de répartition empirique")

#Fonction de répartition théorique
Abs=np.linspace(np.min(Z),np.max(Z),1000)
Ord=[2*x/(1+x) for x in Abs]
plt.plot(Abs,Ord,'b:',label="Fonction de répartition théorique")

plt.legend()
plt.show()
```



Dans la question 1c, nous avons expliqué que ce script trace la fonction qui à t compte la proportion de réalisations de la variable aléatoire qui sont inférieures ou égales à t. Voilà pourquoi on l'appelle la fonction de répartition empirique. D'après la loi faible des grands nombres, elle fournit une approximation de la fonction de répartition (théorique).

Exercice 8 (Fonction de répartition empirique). Soit $(X_n)_{n\geqslant 1}$ une suite de variables aléatoires indépendantes de même loi définies sur un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$. Pour tout $n \in \mathbb{N}^*$, on appelle fonction de répartition empirique de l'échantillon $(X_1, \dots, X_n)$ la fonction

$$F_n : (\omega, t) \in \Omega \times \mathbb{R} \longmapsto \frac{\text{card}(\{i \in [\![1\,;n]\!] \mid X_i(\omega) \leqslant t\})}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{X_i(\omega)\leqslant t}.$$

Autrement dit, pour tous $\omega \in \Omega$ et $t \in \mathbb{R}$, $F_n(\omega, t)$ est la proportion de variables aléatoires de l'échantillon qui sont à valeurs inférieures ou égales à t.

1) À l'aide des exemples de l'exercice précédent, construire une fonction qui prend en argument un vecteur (implémentant un échantillon) et qui trace la courbe de la fonction de répartition empirique de cet échantillon.
2) Pour plusieurs choix de paramètre, tester cette fonction avec un vecteurs de 10000 réalisations de variables aléatoires indépendantes de loi exponentielle et lui superposer la courbe de la fonction de répartition théorique.
3) Pour plusieurs choix de paramètres, tester cette fonction avec un vecteurs de 10000 réalisations de variables aléatoires indépendantes de loi binomiale et lui superposer la courbe de la fonction de répartition théorique.
4) Comment expliquer que ces courbes se superposent ?

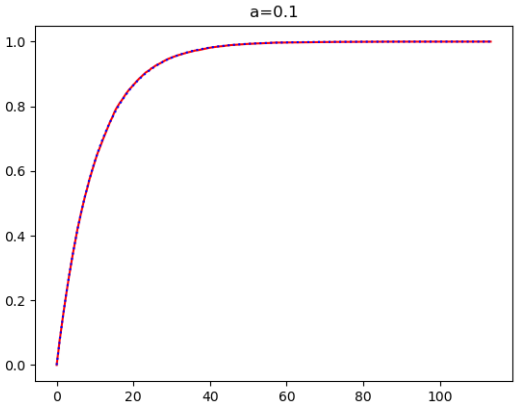
Correction :

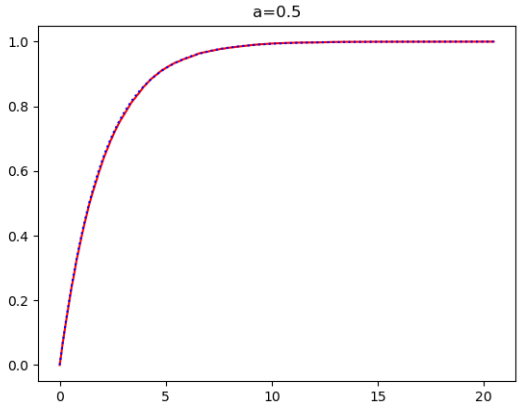
1) Soient `X` une liste en Python et `t` une variable numérique en Python. La commande `X<=t` transforme la liste `X` en une liste de booléens où les coordonnées inférieurs à `t` deviennent `True` et les autres `False`. Ainsi `np.sum(X<=t)` compte le nombre de coordonnées de `X` qui sont inférieures ou égales à `t`.

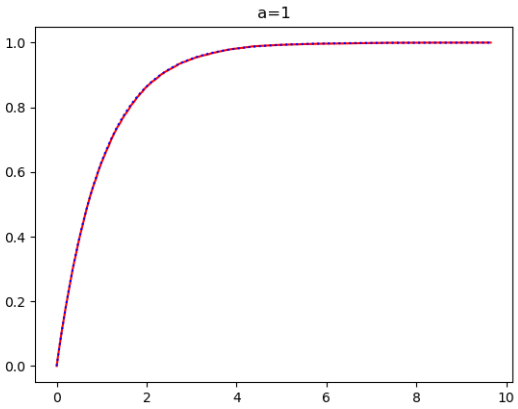
```
import numpy as np
def FoncRepEmp(X):
    a=min(X); b=max(X); n=len(X)
    T=np.linspace(a,b,1000)
    F=[]
    for t in T:
        F.append(np.sum(X<=t)/n)
    plt.plot(T,F,'r',label="Fonction de répartition théorique")
```

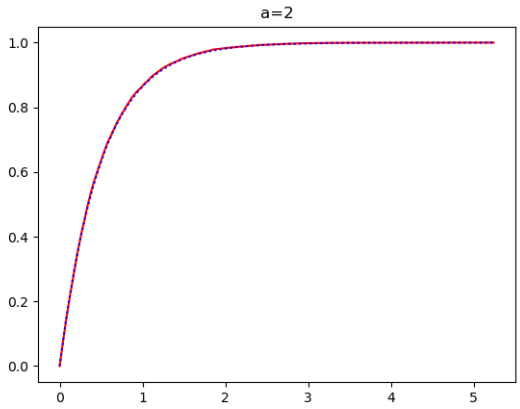
2)

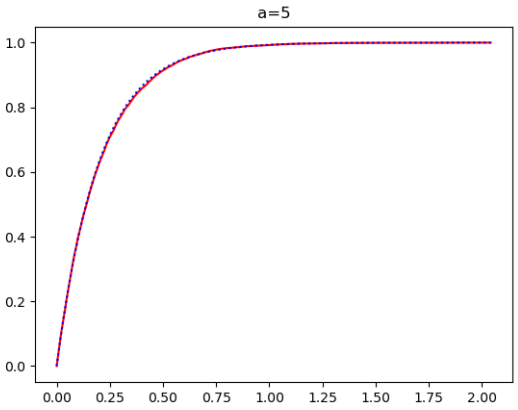
```python
import numpy.random as rd
import matplotlib.pyplot as plt
for a in [0.1,0.5,1,2,5]:
    plt.figure()
    X=rd.exponential(1/a,10000)
    m=max(X)
    Abs=np.linspace(0,m,1000)
    Ord=[1-np.exp(-a*x) for x in Abs]
    FoncRepEmp(X)
    plt.plot(Abs,Ord,'b:',label="Fonction de répartition théorique")
    plt.title('a='+str(a))
    plt.show()
```









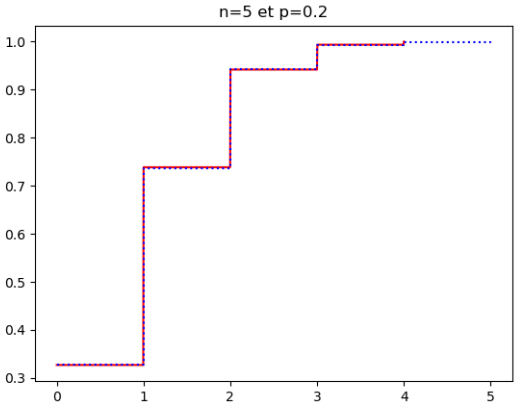


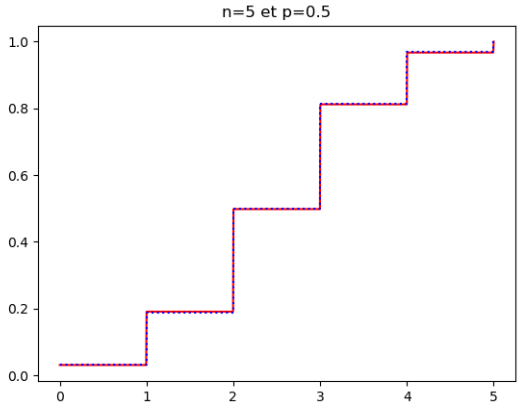
3) Commençons par implémenter la fonction de répartition d'une loi binomiale (nous avons vu comment faire l'an passé) :

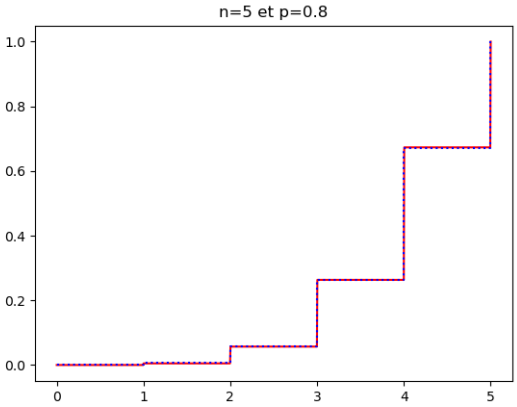
```python
import numpy.random as rd
import matplotlib.pyplot as plt

```

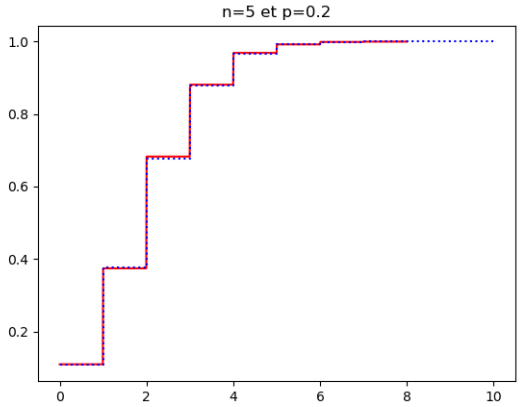
```
def FoncRepBin(n,p):
    x=(1-p)**n
    F=x
    Y=[F]
    for k in range(1,n+1):
        x=(n-k+1)*p*x/(k*(1-p))
        F=F+x
        Y.append(F)
    plt.step(range(n+1),Y,'b:',where="post",label="Fonction de répartition théorique")
```

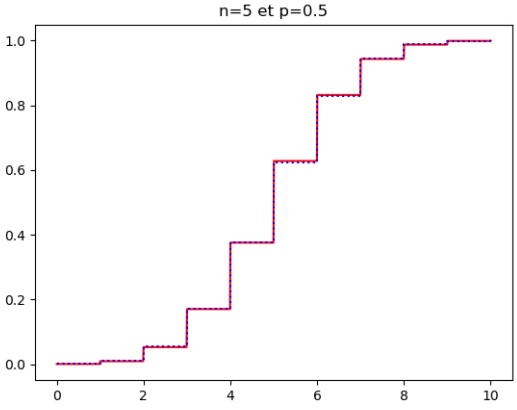
```
for n in [5,10,20]:
    for p in [0.2,0.5,0.8]:
        plt.figure()
        X=rd.binomial(n,p,10000)
        FoncRepEmp(X)
        FoncRepBin(n,p)
        plt.title('n='+str(a)+' et p='+str(p))
plt.show()
```

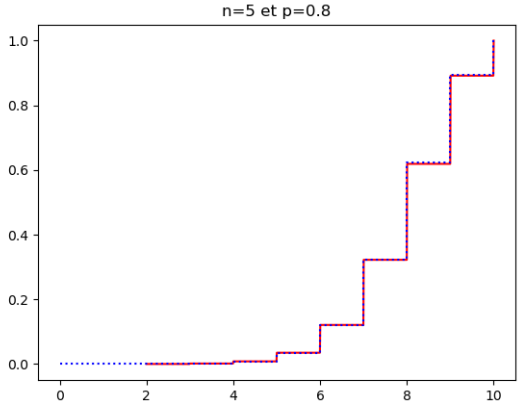


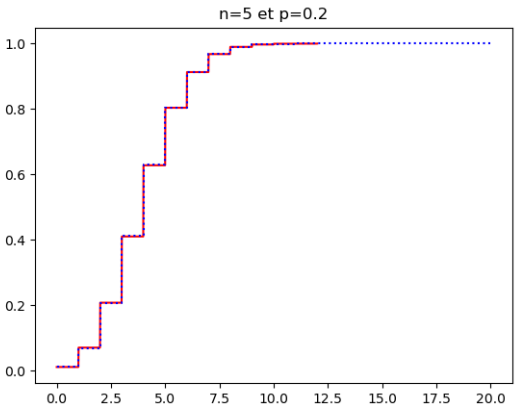


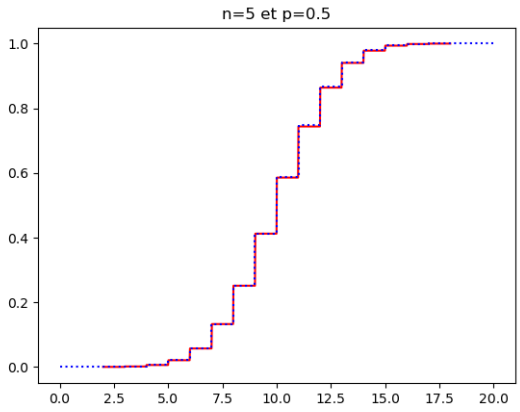


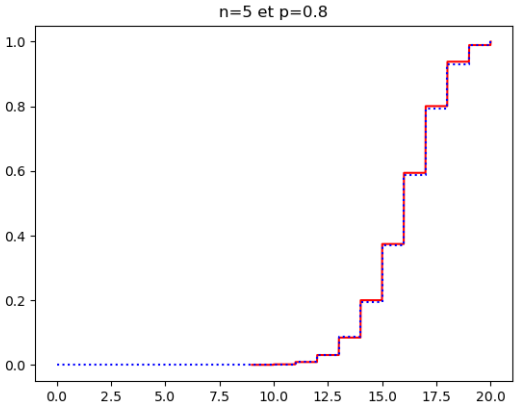












4) Les variables aléatoires X_i, $i \geqslant 1$, sont indépendantes. Soit $t \in \mathbb{R}$. Les variables aléatoires $\mathbb{1}_{X_i \leqslant t}$, $i \geqslant 1$, sont encore indépendantes. Elles ont la même loi et admettent pour espérance $\mathbb{E}(\mathbb{1}_{X_1 \leqslant t}) = \mathbb{P}(X_1 \leqslant t) = F_{X_1}(t)$. Ainsi la loi faible des grands nombres entraîne que

$$\forall \varepsilon > 0, \qquad \mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{X_i \leqslant t} - F_{X_1}(t)\right| \geqslant \varepsilon\right) \xrightarrow[n\to+\infty]{} 0.$$

Autrement dit, pour tout $\omega \in \Omega$, la réalisation $F_n(\omega, t)$ est une $F_{X_1}(t)$. Il est donc attendu que les courbes se superposent.

2) Utilisation de la méthode d'inversion

Exercice 9 (Méthode d'inversion de la fonction de répartition). Soit X une variable aléatoire réelle admettant une densité. On suppose qu'il existe $a \in \mathbb{R} \cup \{-\infty\}$ et $b \in \mathbb{R} \cup \{+\infty\}$ tels que $a < b$ et :

- la fonction de répartition F de X est strictement croissante sur $]a\,;b[$,
- $\mathbb{P}(X \in]a\,;b[) = 1$.

On se donne U une variable aléatoire de loi $\mathcal{U}([0\,;1])$.

1) a) Calculer les limites de F en a et b dans le cas où ce sont des réels.
   b) Justifier que F est une bijection de $]a\,;b[$ sur $]0\,;1[$.
2) a) On pose $Y = F^{-1}(U)$. Déterminer la fonction de répartition de Y.
   b) En déduire une méthode pour simuler la variable aléatoire X.
3) Écrire une fonction qui prend en argument $a > 0$ et qui simule une variable aléatoire de loi $\mathcal{E}(a)$ avec la méthode d'inversion.
4) On dit qu'une variable aléatoire suit une loi de Laplace si elle admet pour densité la fonction $x \longmapsto \frac{1}{2}e^{-|x|}$.

a) Montrer que la fonction de répartition d'une variable de loi de Laplace satisfait aux hypothèses de l'énoncé avec $a = -\infty$ et $b = +\infty$ et que sa bijection réciproque est :

$$x \longmapsto \begin{cases} \ln(2x) & \text{si} \quad x \in]0\,;1/2] \\ -\ln(2(1-x)) & \text{si} \quad x \in]1/2\,;1[\end{cases}$$

b) Écrire une fonction sans argument qui simule une variable aléatoire de loi de Laplace avec la méthode d'inversion.

5) a) Écrire une fonction sans argument et qui simule une variable aléatoire de loi de Cauchy, c'est-à-dire de densité $x \longmapsto \dfrac{1}{\pi(1+x^2)}$.

b) Soit $(X_i)_{i\geqslant 1}$ est une suite de variables aléatoires indépendantes et de loi de Cauchy. Recopier, compléter et exécuter le script suivant afin qu'il représente les 10000 premières valeurs de la suite $\left(\dfrac{1}{n}\displaystyle\sum_{k=1}^{n} X_i\right)_{n\geqslant 1}$.

```
1  ..............................
2  ..............................
3  ..............................
4  L=[]
5  S=0
6  for n in range(1,10001):
7      X=..................#Simule une Cauchy
8      S=...................
9      L.append(...........)
10 ..............................#Trace
11 plt.show()
```

Que remarque-t-on ? Comment l'expliquer ?

6) On se donne maintenant X une variable aléatoire à densité mais on ne suppose plus rien sur sa fonction de répartition. Pour tout $x \in]0\,;1[$, on introduit l'ensemble $I_x = \{t \in \mathbb{R} \,|\, F(t) \geqslant x\}$.

a) Soit $x \in]0\,;1[$. Montrer que I_x admet une borne inférieure. On la note[1] $G(x)$.

b) Montrer que $G(x) \in I_x$ puis que $I_x = [G(x)\,;+\infty[$.
On pourra utiliser, après l'avoir démontré, le fait que $G(x)$ est limite d'une suite d'éléments de I_x.

c) Montrer que, pour tous $x \in]0\,;1[$ et $t \in \mathbb{R}$, $G(x) \leqslant t \iff F(t) \geqslant x$.

d) En déduire que, si $U \hookrightarrow \mathcal{U}(]0\,;1[)$, alors les variables aléatoires X et $G(U)$ ont la même loi.

Correction :

1) a) • Supposons que $a \in \mathbb{R}$. Comme $X \in]a\,;b[$ presque sûrement, si $x \leqslant a$, $F(x) = \mathbb{P}(X \leqslant x) = 0$.
En effet : si $x \leqslant a$, on a $[X \leqslant x] \subset [X \leqslant a] \subset [X \notin I]$. Ainsi $F(x) \leqslant \mathbb{P}(X \notin I) = 0$ et donc $F(x) = 0$.

Nous en déduisons que $\lim\limits_{a^-} F = 0$. Par continuité de F en a (car X est une variable à densité), $\lim\limits_{a} F = 0$.

• Supposons que $b \in \mathbb{R}$. Comme $X \in]a\,;b[$ presque sûrement, si $x \geqslant b$, $F(x) = \mathbb{P}(X \leqslant x) = 1$.
En effet : si $x \geqslant b$, alors $[X \in I] \subset [X \leqslant b] \subset [X \leqslant x]$. Ainsi $1 = \mathbb{P}(X \in I) \leqslant F(x)$ et donc $F(x) = 1$.
Nous en déduisons que $\lim\limits_{b^+} F = 1$. Par continuité de F en b (car X est une variable à densité), $\lim\limits_{b} F = 1$.

b) La fonction F est continue sur $\mathbb{R}$ (car c'est la fonction de répartition d'une variable aléatoire à densité) donc sur $I =]a\,;b[$ et elle est strictement croissante sur I par hypothèse. Le théorème de la bijection entraîne alors que F réalise une bijection de I dans $F(I)$.

D'après le cours, si $b = +\infty$, alors $\lim\limits_{b} F = 1$. Et que, si $a = -\infty$, alors $\lim\limits_{a} F = 0$. Les cas réels ont été traités à la question précédente. Dans tous les cas $F(I) =]0\,;1[$, ce qui permet de conclure.

1. Si $t < G(x)$, alors $t \notin I_x$ donc $F(t) < x$. En faisant tendre t vers $G(x)$, et par continuité de F en $G(x)$ (c'est la fonction de répartition d'une variable à densité) on obtient $F(G(x)) \leqslant x$. Comme $G(x) \in I_x$, on en déduit que $F(G(x)) = x$. Ainsi $G(x)$ est le plus petit antécédent de x. Si F est bijective alors il s'agit de $F^{-1}(x)$. Lorsque F n'est pas bijective, comme elle est croissante, elle possède des paliers et on prend alors la borne inférieure du palier.

2) a) Déjà $\mathbb{P}(Y \in I) = \mathbb{P}(U \in F(I)) = 1$ car $F(I) =]0\,;1[$.
   - Dans le cas où $a \in \mathbb{R}$, $\mathbb{P}(Y \leqslant x) = \mathbb{P}(F^{-1}(U) \leqslant x) = 0$ pour tout $x \leqslant a$.
   - Dans le cas où $b \in \mathbb{R}$, $\mathbb{P}(Y \leqslant x) = \mathbb{P}(F^{-1}(U) \leqslant x) = 1$ pour tout $x \geqslant b$.
   - Supposons que $x \in]a\,;b[$. On a alors $\mathbb{P}(Y \leqslant x) = \mathbb{P}(U \leqslant F(x)) = F_U(F(x)) = F(x)$ puisque $F(x) \in]0\,;1[$ et $F_U(t) = t$ lorsque $t \in [0\,;1]$.

   Ainsi Y admet la même fonction de répartition que X.

   b) On en déduit que X et Y ont la même fonction de répartition et donc la même loi. Ainsi, pour simuler la variable aléatoire X, on simule U et on renvoie $F^{-1}(U)$ (à condition de savoir inverser la fonction F...).

3) Si $X \hookrightarrow \mathcal{E}(a)$, alors $F : t \longmapsto (1-e^{-t})\mathbb{1}_{\mathbb{R}_+}(x)$. Il s'agit bien d'une fonction strictement croissante sur $]0\,;+\infty[$ où $\mathbb{P}(X \in]0\,;+\infty[) = 1$. Si $x \in [0\,;1]$ et $t \in]0\,;+\infty[$,

$$x = F(t) \quad \iff \quad 1 - x = e^{-at} \quad \iff \quad t = -\frac{1}{a}\ln(1-x).$$

Ainsi $F^{-1} : x \longmapsto -\dfrac{1}{a}\ln(1-x)$. D'où la fonction :

```
import numpy as np
import numpy.random as rd
def SimulExp(a):
    U=rd.random()
    return -np.log(1-U)/a
```

4) a) Supposons que X est une variable aléatoire de loi de Laplace. Si $t \in \mathbb{R}_-$, alors

$$F(t) = \int_{-\infty}^{t} \frac{1}{2}e^{-|x|}\,dx = \frac{1}{2}\int_{-\infty}^{t} e^{x}\,dx = \frac{1}{2}\lim_{A\to-\infty}[e^x]_A^t = \frac{1}{2}e^t.$$

Si $t \in \mathbb{R}_+^*$, alors

$$F(t) = \int_{-\infty}^{0} \frac{1}{2}e^{-|x|}\,dx + \int_0^t \frac{1}{2}e^{-|x|}\,dx = \frac{1}{2} + \frac{1}{2}\int_0^t e^{-x}\,dx.$$

La valeur $1/2$ pour la première intégrale consiste à remarquer que c'est l'intégrale du cas $t \in \mathbb{R}_-$ dans le cas particulier où $t = 0$. Ensuite

$$F(t) = \frac{1}{2} + \frac{1}{2}\lim_{A\to+\infty}\left[-e^{-x}\right]_0^t = \frac{1}{2} + \frac{1}{2}(-e^{-t}+1) = 1 - \frac{1}{2}e^{-t}.$$

La fonction F est bien strictement croissante sur $\mathbb{R}$.

- Soit $t \in \mathbb{R}_-$ et $x \in\in]0\,;1/2]$. On a

$$x = F(t) \quad \iff \quad x = \frac{1}{2}e^t \quad \iff \quad t = \ln(2x).$$

- Soit $t \in \mathbb{R}_+^*$ et $x \in\in]1/2\,;1[$. On a

$$x = F(t) \quad \iff \quad 1 - x = \frac{1}{2}e^{-t} \quad \iff \quad t = -\ln(2(1-x)).$$

Ainsi

$$F^{-1} : x \longmapsto \begin{cases} \ln(2x) & \text{si} \quad x \in]0\,;1/2] \\ -\ln(2(1-x)) & \text{si} \quad x \in]1/2\,;1[\end{cases}$$

b)

```
import numpy as np
import numpy.random as rd
def SimulLaplace():
    U=rd.random()
    if U<=1/2:
        return -np.log(2*U)
    else:
        return -np.log(2*(1-U))
```

5) a) Si X suit une loi de Cauchy alors, pour tout $x \in \mathbb{R}$,

$$F(t) = \int_{-\infty}^{t} \frac{1}{\pi(1+x^2)}\,dx = \lim_{B\to-\infty}\left[\frac{1}{\pi}\operatorname{Arctan}(x)\right]_B^t = \frac{1}{\pi}\operatorname{Arctan}(t) + \frac{1}{2}.$$

Il s'agit bien d'une fonction strictement croissante sur $I = \mathbb{R}$. Si $x \in [0\,;1]$ et $t \in \mathbb{R}$,

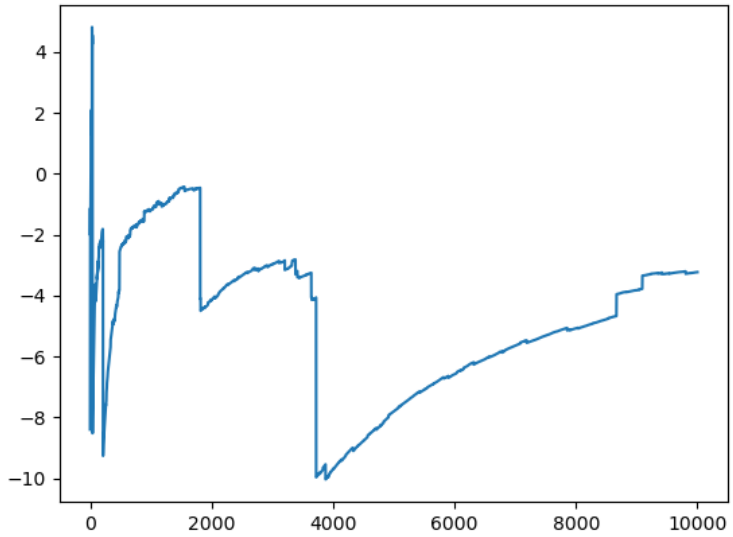
$$x = F(t) \iff x - \frac{1}{2} = \frac{1}{\pi}\operatorname{Arctan}(t) \iff t = \tan\left(\pi x - \frac{\pi}{2}\right).$$

Ainsi $F^{-1} : x \longmapsto \tan\left(\pi x - \frac{\pi}{2}\right)$. D'où la fonction :

```
import numpy as np
import numpy.random as rd
def SimulCauchy():
    U=rd.random()
    return np.tan(np.pi*(U-1/2))
```

b)

```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
L=[]
S=0
for n in range(1,10001):
    X=SimulCauchy()#Simule une Cauchy
    S=S+X
    L.append(S/n)
plt.plot(range(1,10001),L)
plt.show()
```



On remarque qu'il n'y a pas de convergence. Or on pourrait s'attendre à ce que $\left(\frac{1}{n}\sum_{k=1}^{n} X_i\right)_{n\geqslant 1}$ converge vers l'espérance de X. En fait ici X n'admet pas d'espérance puisque

$$xf(x) = \frac{x}{\pi(1+x^2)} \sim \frac{1}{\pi x}.$$

et qu'il ne s'agit pas d'une fonction donc l'intégrale converge sur $\mathbb{R}$.

6) a) On a $F(t) \xrightarrow[t\to+\infty]{} 1$ donc, comme $x < 1$, il existe $A > 0$ tel que, pour tout $t \geqslant A$, $F(t) \geqslant x$. Ainsi I_x est non vide (il contient A).
On a $F(t) \xrightarrow[t\to-\infty]{} 0$ donc, comme $x > 0$, il existe $B < 0$ tel que, pour tout $t \leqslant B$, $F(t) < x$. Ainsi I_x est minorée (par B).
Le théorème de la borne inférieure garantit alors l'existence de la borne supérieure de I_x.

b) Soit $n \in \mathbb{N}^*$. Puisque $G(x)+\dfrac{1}{n} > G(x)$, il existe $t_n \in I_x$ tel que $G(x) \leqslant t_n \leqslant G(x)+\dfrac{1}{n}$. Par encadrement, on a donc $t_n \xrightarrow[n\to+\infty]{} G(x)$.

Pour tout $n \in \mathbb{N}^*$, $t_n \in I_x$ donc $F(t_n) \geqslant x$. Comme $t_n \xrightarrow[n\to+\infty]{} G(x)$, par continuité de F en $G(x)$ (c'est une variable à densité), on a obtient alors $F(G(x)) \geqslant x$. Autrement dit $G(x) \in I_x$.

Par définition de $G(x)$, on a $I_x \subset [G(x)\,;+\infty[$. Réciproquement, donnons-nous $y \in [G(x)\,;+\infty[$. Comme G est croissante, on a $F(y) \geqslant F(G(x)) \geqslant x$ donc $y \in I_x$. Ainsi $I_x = [G(x)\,;+\infty[$.

c) Soient $x \in]0\,;1[$ et $t \in \mathbb{R}$. On a $G(x) \leqslant t$ si et seulement si $t \in [G(x)\,;+\infty[$ si et seulement si $t \in I_x$ si et seulement si $F(t) \geqslant x$.

d) On a donc, pour tout $x \in \mathbb{R}$,

$$\mathbb{P}(G(U) \leqslant t) = \mathbb{P}(U \leqslant F(t) \geqslant U) = F_U(F(t)) = F(t).$$

Ainsi X et $G(U)$ ont bien la même loi.

3) Retour sur la simulation de lois discrètes

Exercice 10 (Lois géométriques à partir de lois exponentielles).

1) Soit $X_a \hookrightarrow \mathcal{E}(a)$ avec $a > 0$. Déterminer la loi de $Y_a = \lfloor X_a \rfloor + 1$.
2) En déduire une méthode de simulation d'une variable aléatoire de loi géométrique de paramètre $p \in]0\,;1[$ en utilisant une simulation d'une variable aléatoire de loi exponentielle (à l'aide de la méthode d'inversion par exemple).
3) Écrire une fonction qui prend en argument $p \in]0\,;1[$, qui trace le diagramme à barres empirique d'un échantillon de 10000 simulations de lois géométriques obtenues via cette méthode, et lui superpose le diagramme à barre théorique d'une loi $\mathcal{G}(p)$. La tester pour différentes valeurs de p.

Correction :

1) Déjà $X_a(\Omega) = \mathbb{R}_+^*$ donc $\lfloor X_a \rfloor(\Omega) = \mathbb{N}$ donc $Y_a(\Omega) = \mathbb{N}^*$. Soit $k \in \mathbb{N}^*$. On a

$$\begin{aligned}\mathbb{P}(Y_a = k) = \mathbb{P}(\lfloor X_a \rfloor = k-1) = \mathbb{P}(k-1 \leqslant X_a < k) = F_{X_a}(k) - F_{X_a}(k-1) &= 1 - e^{-ak} - (1 - e^{-a(k-1)}) \\ &= e^{-a(k-1)} - e^{-ak} \\ &= e^{-a(k-1)}(1 - e^{-a}) \\ &= (1-p_a)^{k-1} p_a\end{aligned}$$

avec $p_a = 1 - e^{-a}$. Ainsi $Y_a \hookrightarrow \mathcal{G}(p_a)$.

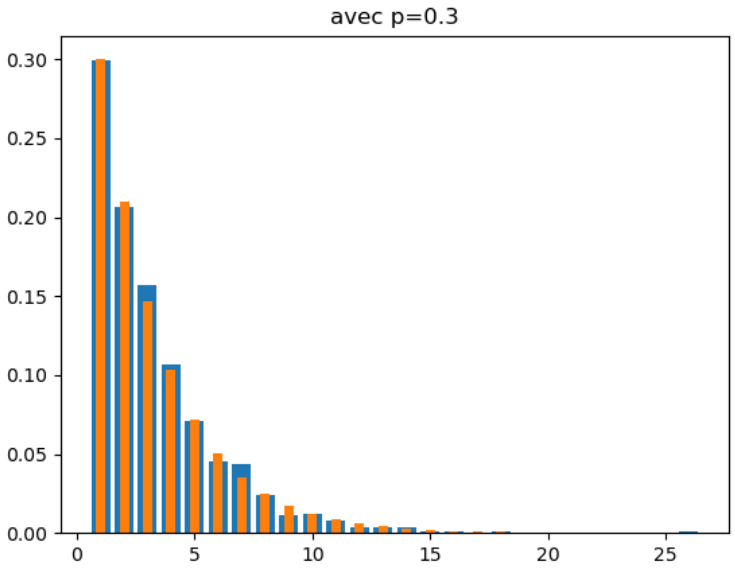
2) Soient $p \in]0\,;1[$ et $a > 0$. On a $p = 1 - e^{-a}$ si et seulement si $a = -\ln(1-p)$. D'où la fonction :

```
import numpy as np
import numpy.random as rd
def SimulGeom(p):
    U=rd.random()
    a=-np.log(1-p)
    X=-np.log(1-U)/a#Loi exp avec la méthode d'inversion
    return int(X)+1
```

3)

```
import matplotlib.pyplot as plt
def DiagBarre(p):
    #Le diagramme à barre empirique
    Y=[SimulGeom(p) for k in range(1000)]#1000 réalisations de
    SimulGoem
    m=max(Y)
    L=np.zeros(m)
    for y in Y:
        L[y-1]=L[y-1]+1#Attention au décalage des indices (la
```

```
    valeur 1 en indice 0, etc.)
    L=L/1000
    plt.bar(range(1,m+1),L)
    #Le diagramme à barre théorique
    T=[p*(1-p)**(k-1) for k in range(1,m+1)]
    plt.bar(range(1,m+1),T,0.4)
    plt.title("avec p="+str(p))
    plt.show()
```



Exercice 11 (Simulation de lois de Poisson). Soit $\lambda \in \mathbb{R}_+^*$. On se donne $(U_i)_{i\geqslant 1}$ une suite de variables aléatoires indépendantes et de même loi $\mathcal{U}(]0\,;1])$. On considère la variable aléatoire

$$Y = \min\{n \in \mathbb{N} \,|\, U_1 U_2 \dots U_{n+1} \leqslant e^{-\lambda}\}.$$

1) Pour tout $i \in \mathbb{N}^*$, on pose $F_i = -\ln(U_i)$. Quelle est la loi de F_i ?
2) En déduire la loi de $S_n = F_1 + \dots + F_n$ pour tout $n \in \mathbb{N}^*$.
3) Calculer $\mathbb{P}(Y > n)$ pour tout $n \in \mathbb{N}^*$.
4) En déduire que Y suit une loi de Poisson de paramètre λ. Commenter en réécrivant Y à l'aide des F_i, $i \geqslant 1$.
5) Compléter la fonction suivante afin qu'elle prenne en argument λ et renvoie la valeur d'une réalisation d'une variable aléatoire de loi $\mathcal{P}(\lambda)$.

```
.............................
def Poisson(lam):
    n=0
    P=rd.random()
    while ..............
        ............
    return n
```

Correction :

1) $U_i(\Omega) =]0\,;1]$ donc $\ln(U_i)(\Omega) =]-\infty\,;0]$ et donc $F_i(\Omega) = \mathbb{R}_+^*$. Soit $t \in \mathbb{R}_+^*$. On a

$$\mathbb{P}(F_i \leqslant t) = \mathbb{P}(\ln(U_i) \geqslant -t) = \mathbb{P}(U_i \geqslant e^{-t}) = 1 - \mathbb{P}(U_i < e^{-t}).$$

Comme $x = e^{-t} \in]0\,;1]$, $\mathbb{P}(U_i < x) = x$ donc $\mathbb{P}(F_i \leqslant t) = 1 - e^{-t}$. Ainsi $F_i \hookrightarrow \mathcal{E}(1)$.

2) Soit $n \in \mathbb{N}^*$. Puisque $F_1, \dots, F_n$ sont indépendantes de même loi $\mathcal{E}(1)$, leur somme S_n suit une loi $\gamma(n)$.

3) Commençons par remarquer que, comme les U_i, $i \geqslant 1$ sont à valeurs dans $]0\,;1]$, la suite $(U_1U_2\dots U_{n+1})_{n\in\mathbb{N}}$ est décroissante. Soit $n \in \mathbb{N}$, $[Y > n]$ est réalisé si et seulement si le plus petit $i \in \mathbb{N}$ tel que $[U_1U_2\dots U_{i+1} \leqslant e^{-\lambda}]$ est réalisé est supérieur strictement à n si et seulement si $[U_1U_2\dots U_{n+1} \leqslant e^{-\lambda}]$ n'est pas réalisé. Ainsi

$$\mathbb{P}(Y > n) = \mathbb{P}(U_1U_2\dots U_{n+1} > e^{-\lambda}) = \mathbb{P}(\ln(U_1U_2\dots U_{n+1}) > -\lambda) = \mathbb{P}(-S_{n+1} > -\lambda) = \mathbb{P}(S_{n+1} < \lambda)$$

et donc $\mathbb{P}(Y > n) = \displaystyle\int_0^\lambda \frac{t^n e^{-t}}{\Gamma(n+1)}\,dt = \int_0^\lambda \frac{t^n e^{-t}}{n!}\,dt$.

Cherchons une relation de récurrence pour la suite $(\mathbb{P}(Y > n))_{n\in\mathbb{N}}$. Fixons $n \in \mathbb{N}^*$. Réalisons une IPP avec les fonctions $u : t \longmapsto \dfrac{t^n}{n!}$ et $v : t \longmapsto -e^{-t}$ de classe C^1 sur $\mathbb{R}_+$. On a $u' : t \longmapsto \dfrac{t^{n-1}}{(n-1)!}$ et $v' : t \longmapsto e^{-t}$ et

$$\int_0^\lambda \frac{t^n e^{-t}}{n!}\,dt = \left[-\frac{t^n}{n!}e^{-t}\right]_0^\lambda - \int_0^\lambda -\frac{t^{n-1}e^{-t}}{(n-1)!}\,dt = \int_0^\lambda \frac{t^n e^{-t}}{n!}\,dt = -\frac{\lambda^n}{n!}e^{-\lambda} + \int_0^\lambda \frac{t^{n-1}e^{-t}}{(n-1)!}\,dt.$$

On a donc

$$\mathbb{P}(Y > n) = -\frac{\lambda^n}{n!}e^{-\lambda} + \mathbb{P}(Y > n-1).$$

Ainsi

$$\mathbb{P}(Y > n) - \mathbb{P}(Y > n-1) = -\frac{\lambda^n}{n!}e^{-\lambda}.$$

On en déduit que, pour tout $n \in \mathbb{N}$,

$$\mathbb{P}(Y > n) - \mathbb{P}(Y > 0) = \sum_{k=1}^{n}(\mathbb{P}(Y > k) - \mathbb{P}(Y > k-1)) = -\sum_{k=1}^{n}\frac{\lambda^k}{k!}e^{-\lambda}.$$

(on reconnaît une somme télescopique). On a

$$\mathbb{P}(Y > 0) = \int_0^\lambda e^{-t}\,dt = 1 - e^{-\lambda} = 1 - \frac{t^0}{0!}e^{-\lambda}.$$

Ainsi

$$\mathbb{P}(Y > n) = 1 - \sum_{k=0}^{n}\frac{\lambda^k}{k!}e^{-\lambda}.$$

4) Pour tout $n \in \mathbb{N}$

$$\mathbb{P}(Y = n) = \mathbb{P}(Y > n-1) - \mathbb{P}(Y > n) = \frac{\lambda^n}{n!}e^{-\lambda}.$$

Ainsi Y suit une loi de Poisson de paramètre λ.

Pour tout $i \in \mathbb{N}$, F_i suit une loi exponentielle de paramètre 1 donc représente une durée de vie ou un temps d'attente de réalisation d'un phénomène sans mémoire. Ainsi, pour tout $n \in \mathbb{N}$, $F_1 + \dots + F_n$ est le temps d'attente de la réalisation de n de ces phénomènes consécutivement et indépendamment. Comme $\ln$ est strictement croissante sur $\mathbb{R}_+^*$, on a

$$Y = \min\{n \in \mathbb{N} \mid \ln(U_1U_2\dots U_{n+1}) \leqslant -\lambda\} = \min\{n \in \mathbb{N} \mid F_1 + F_2 + \dots + F_{n+1} \geqslant \lambda\}.$$

Ainsi, pour tout $n \in \mathbb{N}$, Y prend la valeur n si et seulement si $F_1+F_2+\dots+F_n < \lambda$ et $F_1+F_2+\dots+F_n \geqslant \lambda$ si et seulement si exactement n phénomènes se sont réalisés pendant la durée λ. On retrouve bien l'interprétation d'une loi de Poisson comme le nombre d'occurrence d'un phénomène dans une durée donnée.

5)

```
import numpy.random as rd
def Poisson(lam):
    n=0
    P=rd.random()
    while P<np.exp(-lam):
        P=P*rd.random()
        n=n+1
    return n
```

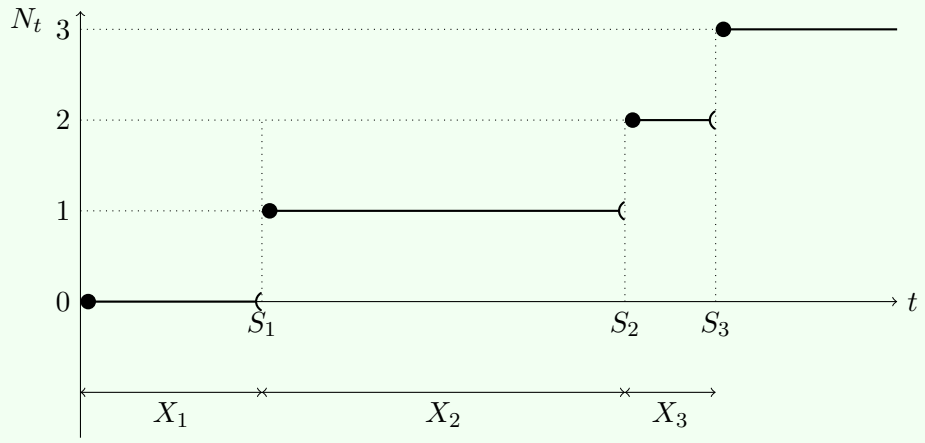
Exercice 12 (D'après concours). On considère une suite de variables aléatoires $(X_n)_{n\geqslant 1}$, définies sur un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$ et qui sont mutuellement indépendantes et identiquement distribuées selon une loi exponentielle de paramètre $\lambda \in \mathbb{R}_+^*$. Pour tout $n \in \mathbb{N}$, on note $S_n = \sum_{k=1}^{n} X_k$, avec la convention $S_0 = 0$.

Enfin, pour tout $t \in \mathbb{R}_+^*$, on note N_t la variable aléatoire égale à la plus grande valeur de n pour laquelle S_n est inférieure ou égale à t, c'est-à-dire :

$$\forall \omega \in \Omega, \qquad N_t(\omega) = \sup\{n \in \mathbb{N} \,|\, S_n(\omega) \leqslant t\}.$$

Par convention, si l'ensemble écrit ci-dessus n'est pas fini, N_t prend la valeur -1.

Voici un exemple de réalisation de N_t en fonction de t :



1) Pour tout $t \in \mathbb{R}_+^*$, montrer que $\mathbb{P}(N_t = 0) = e^{-\lambda t}$.
2) Pour tout $n \in \mathbb{N}^*$, déterminer une densité de la variable aléatoire λS_n.
3) Soit $t \in \mathbb{R}_+^*$.
   a) Comparer les événements $[N_t = -1]$ et $\bigcap_{n=1}^{+\infty} [S_n \leqslant t]$.
   b) Montrer que, pour tout $x \in \mathbb{R}_+^*$,
   $$\lim_{n\to+\infty} \int_0^x \frac{u^{n-1}}{(n-1)!} e^{-u}\, du = 0.$$
   c) En déduire que $\mathbb{P}(N_t = -1) = 0$.
   d) Établir que, pour tout $n \in \mathbb{N}$, $\mathbb{P}(N_t \geqslant n) = \mathbb{P}(S_n \leqslant t)$.
4) Soient $t \in \mathbb{R}_+^*$ et $n \in \mathbb{N}$.
   a) Montrer que $\mathbb{P}([N_t = n]) = \mathbb{P}([S_n \leqslant t] \cap [S_{n+1} > t])$.
   b) En déduire que
   $$\mathbb{P}(N_t = n) = \int_0^{\lambda t} \frac{u^{n-1}}{(n-1)!} e^{-u}\, du - \int_0^{\lambda t} \frac{u^n}{n!} e^{-u}\, du.$$
   c) En intégrant par parties une des intégrales ci-dessus, montrer que
   $$\forall n \in \mathbb{N}^*, \qquad \mathbb{P}(N_t = n) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}.$$
   Quelle est la loi de N_t?
5) a) Écrire une fonction Python d'en-tête `def simulation_S(n,lam):` renvoyant une réalisation de S_n.
   b) Écrire une fonction Python d'en-tête `def simulation_N(t,lam):` renvoyant une réalisation de N_t.

c) Compléter la fonction en Python `evolution_S` afin qu'elle renvoie toutes les valeurs $S_1, S_2, \dots, S_n$ tant que $S_n \leqslant t$.

```
.....
def evolution_S(t,lam):
    L=[]
    S=.....
    while .....
        L.append(S)
        S=S+.....
    return .....
```

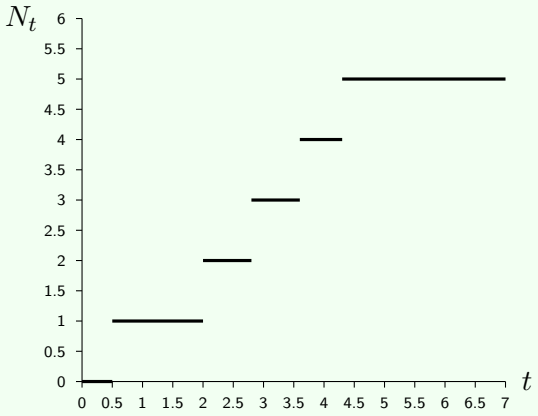
d) On a commencé à écrire une fonction Python ci-dessous. Dans ce script, on implémente par `S` la liste constituée de $S_1, \dots, S_n$ (où n est le plus grand entier tel que $S_n \leqslant T$) et on souhaite tracer l'évolution de N_t du temps $t = 0$ au temps $t = T$ de la même manière que sur la figure de la page précédente.

```
import matplotlib.pyplot as plt
def trace_N(T,lam):
    S=evolution_S(T,lam)
    n=len(S)
    plt.plot([0,S[0]],[0,0])
    for i in range(1,n):
        .....
    plt.show()
```

Par laquelle des instructions suivantes faut-il compléter la ligne manquante ?

(i) `plt.plot([i-1,i],[S[i-1],S[i-1]])` (ii) `plt.plot([i,i+1],[S[i],S[i+1]])`
(iii) `plt.plot([S[i],S[i+1]],[i,i])` (iv) `plt.plot([S[i-1],S[i]],[i,i])`

e) Un étudiant exécute le script précédent pour $T = 7$ et $\lambda = 1$ et obtient la figure suivante :



Que valent dans ce cas $N_{3,2}$ et $N_{5,5}$? Donner une valeur approximative de S_2 et de X_4.

Correction : *En fait c'est presque le même exercice que le précédent mais formulé différemment (et l'on montre que le* min *(qui est un* sup *ici) existe bien presque sûrement).*

1) Pour tout $t \in \mathbb{R}_+^*$, $[N_t = 0]$ est réalisé si et seulement si $[S_1 > t]$ est réalisé. Ainsi

$$\mathbb{P}(N_t = 0) = \mathbb{P}(S_1 > t) = 1 - \mathbb{P}(S_1 \leqslant t) = 1 - (1 - e^{-\lambda t}) = e^{-\lambda t}.$$

2) Soit $n \in \mathbb{N}^*$. Pour tout $k \in [\![1\,;n]\!]$, $X_k \hookrightarrow \mathcal{E}(\lambda)$ donc $\lambda X_k \hookrightarrow \mathcal{E}(1)$. Ainsi λS_n est la somme de n variables aléatoires indépendantes de loi $\mathcal{E}(1)$. Par conséquent $\lambda S_n \hookrightarrow \gamma(n)$ et admet donc pour densité

$$t \longmapsto \frac{t^{n-1}}{(n-1)!} e^{-t} \mathbb{1}_{t \geqslant 0}.$$

3) a) La variable aléatoire N_t prend la valeur -1 si et seulement si la réalisation de l'ensemble $\{n \in \mathbb{N} \,|\, S_n \leqslant t\}$ n'est pas un ensemble fini, c'est-à-dire si et seulement si, pour tout $n \in \mathbb{N}$, $[S_n \leqslant t]$ est réalisé. Ainsi $[N_t = -1] = \bigcap_{n=1}^{+\infty} [S_n \leqslant t]$.

b) Soit $x \in \mathbb{R}_+^*$. Pour tout $u \in [0\,;x]$, $e^{-u} \leqslant 1$ donc $0 \leqslant \dfrac{u^{n-1}}{(n-1)!}e^{-u} \leqslant \dfrac{u^{n-1}}{(n-1)!}$. Ainsi par propriété de croissance des intégrales de fonctions continues,

$$0 \leqslant \int_0^x \frac{u^{n-1}}{(n-1)!}e^{-u}\,du \leqslant \int_0^x \frac{u^{n-1}}{(n-1)!}\,du = \left[\frac{u^n}{n!}\right]_0^x = \frac{x^n}{n!}.$$

Par croissances comparées, on a $\dfrac{x^n}{n!} \xrightarrow[n\to+\infty]{} 0$. Par théorème d'encadrement, on en déduit que

$$\lim_{n\to+\infty} \int_0^x \frac{u^{n-1}}{(n-1)!}e^{-u}\,du = 0.$$

c) La suite $([S_n \leqslant t])_{n\in\mathbb{N}}$ est décroissante (en effet, pour tout $n \in \mathbb{N}$, si $S_{n+1} \leqslant t$, alors $S_n = S_{n+1} - X_{n+1} \leqslant t$ puisque $X_{n+1}(\Omega) = \mathbb{R}_+^*$). Par propriété de la limite monotone,

$$\mathbb{P}(N_t = 1) = \mathbb{P}\left(\bigcap_{n=1}^{+\infty}[S_n \leqslant t]\right) = \lim_{n\to+\infty} \mathbb{P}(S_n \leqslant t) = \lim_{n\to+\infty} \int_0^t \frac{u^{n-1}}{(n-1)!}e^{-u}\,du.$$

D'après la question précédente, $\mathbb{P}(N_t = -1) = 0$.

d) Pour tout $n \in \mathbb{N}$, $[N_t \geqslant n]$ est réalisé si et seulement si le sup de la définition est réalisé après l'instant n si et seulement si, à l'instant n, S_n prend encore une valeur inférieure ou égale à t. Ainsi $[N_t \geqslant n] = [S_n \leqslant t]$ et donc $\mathbb{P}(N_t \geqslant n) = \mathbb{P}(S_n \leqslant t)$.

4) a) On a

$$\mathbb{P}(N_t = n) = \mathbb{P}(N_t \geqslant n) - \mathbb{P}(N_t \geqslant n+1) = \mathbb{P}(S_n \leqslant t) - \mathbb{P}(S_{n+1} \leqslant t)$$

et donc $\mathbb{P}([N_t = n]) = \mathbb{P}([S_n \leqslant t] \cap [S_{n+1} > t])$.

b) Ainsi $\mathbb{P}([N_t = n]) = \mathbb{P}(\lambda S_n \leqslant \lambda t) - \mathbb{P}(\lambda S_{n+1} \leqslant \lambda t)$. D'après la question 2, on obtient

$$\mathbb{P}\left(N_t = n\right) = \int_0^{\lambda t} \frac{u^{n-1}}{(n-1)!}e^{-u}\,du - \int_0^{\lambda t} \frac{u^n}{n!}e^{-u}\,du.$$

c) C'est la même IPP que dans l'exercice précédent :

$$\mathbb{P}\left(N_t = n\right) = \frac{(\lambda t)^n}{n!}e^{-\lambda t}.$$

Ainsi $N_t \hookrightarrow \mathcal{P}(\lambda t)$.

5) a)

```
import numpy.random as rd
def simulation_S(n,lam):
    S=0
    for k in range(n):
        S=S+rd.exponential(1/lam)
    return S
```

b)

```
import numpy.random as rd
def simulation_N(t,lam):
    n=0
    S=rd.exponential(1/lam)
    while S<=t:
        n=n+1
        S=S+rd.exponential(1/lam)
    return n
```

c) Compléter la fonction en Python `evolution_S` afin qu'elle renvoie toutes les valeurs $S_1, S_2, \ldots, S_n$ tant que $S_n \leqslant t$.

```
import numpy.random as rd
def evolution_S(t,lam):
    L=[]
    S=rd.exponential(1/lam)
    while S<=t:
        L.append(S)
        S=S+rd.exponential(1/lam)
    return L
```

d) Pour tout $i \in \mathbb{N}$, entre les instants S_i (implémenté en indice `i-1`) et S_{i+1} (implémenté en indice `i`), $t \longmapsto N_t$ est constante égale à i. Ainsi il faut ajouter`plt.plot([S[i-1],S[i]],[i,i])`.

e) On lit que $N_{3,2} = 3$ et $N_{5,5} = 5$. Il semble que $S_2 \approx 2$, $S_3 \approx 2,8$ et $S_4 \approx 3,6$ si bien que $X_4 = S_4 - S_3 \approx 3,6 - 2,8 = 0,8$.

Exercice 13 (Simulation de lois finies quelconques). Soit X une variable aléatoire finie définie sur un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$. On suppose que $X(\Omega) = \{x_1, \dots, x_n\}$ et, pour tout $i \in [\![1\,;n]\!]$, on note $p_i = \mathbb{P}(X = x_i)$. Pour tout $k \in [\![1\,;n]\!]$, notons $f_k = \sum_{i=1}^{k} p_i$. On pose $f_0 = 0$.

1) Soit U une variable aléatoire de loi $\mathcal{U}(]0\,;1[)$. Montrer que, pour tout $k \in [\![1\,;n]\!]$, $\mathbb{P}(f_{k-1} < U \leqslant f_k) = p_k$.

2) En déduire une méthode de simulation de X à l'aide de U.

3) Soit `P` une liste implémentée en Python. Que font les commandes suivantes ?

```
F=np.cumsum(P)
np.sum(F<rd.random())
```

4) En déduire une fonction qui prend en entrée une liste implémentant $x_1, \dots, x_n$ et une liste implémentant $p_1, \dots, p_n$ et qui simule X.

Correction :

1) Soit $k \in [\![1\,;n]\!]$. On a $f_k \in [0\,;1]$ et $f_{k-1} \in [0\,;1]$ donc

$$\mathbb{P}(f_{k-1} < U \leqslant f_k) = \mathbb{P}(U \leqslant f_k) - \mathbb{P}(U \leqslant f_{k-1}) = f_k - f_{k-1} = p_k.$$

2) Pour simuler X, il suffit de simuler une loi uniforme sur $]0\,;1[$ et chercher $k \in [\![1\,;k]\!]$ tel que cette réalisation tombe entre f_{k-1} et f_k (ce qui arrive bien avec probabilité p_k d'après la question précédente).

3) Si `P` est une liste qui implémente $p_1, p_2, \dots, p_n$, alors la variable `F` est un tableau unidimensionnel qui contient $p_1 = f_1$, $p_1 + p_2 = f_2$, $p_1 + p_2 + p_3 = f_3$, etc. $p_1 + p_2 + \cdots + p_n = f_n = 1$. La commande `np.sum(F<rd.random())` compte le nombre de valeurs de la liste `F` qui sont inférieures à la réalisation d'une variable aléatoire de loi uniforme sur $]0\,;1[$. Comme $(f_k)_{k \in [\![1\,;n]\!]}$ est croissante, cette variable renvoie donc l'entier k voulu.

4)
```
def SimuleP(X,P):
    F=np.cumsum(P)
    k=np.sum(F<rd.random())
    return X[k-1]
```