Python 2
Simulations d'expériences aléatoires

L'objectif est l'utilisation de **Python** dans la modélisation d'expériences et variables aléatoires. Nous aurons besoin de la bibliothèque `numpy.random` à importer ainsi :

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

Nous aurons également besoin, pour les graphiques, d'importer :

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

Pour représenter un histogramme de données contenues dans la liste `Lvaleurs` en fonction de la listes de bornes `Lbornes`, on utilise :

```
plt.hist(Lvaleurs,Lbornes)
```

Petite remarque
Par défaut, il s'agit d'un histogramme d'effectifs. Pour obtenir un histogramme de fréquences, on ajoute `density=True` en option dans la commande `plt.hist()`.

Après la création de l'histogramme, il faut rajouter la ligne `plt.show()` pour le représenter.
Nous aurons besoin de la commande `rd.random()` qui renvoie un réel aléatoire de $]0;1[$

I Simuler les lois binomiale et géométrique avec random

Que permet de faire le programme suivant ?

```
1 import numpy.random as rd
2
3 def bernoulli(p):
4     r=rd.random()
5     if r<p:
6         return 1
7     else :
8         return 0
```

Ce programme renvoie une réalisation d'une variable aléatoire suivant la loi de Bernoulli de paramètre p.
En effet, un réel r est choisi aléatoirement dans $]0;1[$ (en ligne 4). Puis :

- si ce réel est inférieur à p, le programme renvoie 1 ;
- sinon, le programme renvoie 0.

Or, la probabilité pour qu'un réel de $]0;1[$ soit inférieur à un réel p (avec $p \in]0;1[$) est égale à p.

Conclusion : l'exécution de `bernoulli(p)` renvoie une réalisation d'une variable aléatoire suivant la loi de Bernoulli de paramètre p.

Vocabulaire
- **Simuler une expérience**.
- Renvoyer/simuler **une réalisation d'une variable aléatoire**.

I.1 Loi binomiale

1. En utilisant la fonction donnée ci-dessus, écrire une fonction `liste_bernoulli` prenant en arguments d'entrée un entier naturel n et un réel p de $]0;1[$ et renvoyant une liste de n réalisations indépendantes d'une variable aléatoire suivant une loi $\mathscr{B}(p)$.
2. En déduire une fonction `binomiale` prenant en arguments d'entrée un entier naturel n et un réel p de $]0;1[$ et renvoyant une réalisation d'une variable aléatoire suivant la loi $\mathscr{B}(n;p)$.
3. Proposer une autre fonction permettant de simuler une réalisation d'une variable aléatoire suivant la loi $\mathscr{B}(n;p)$ qui n'utilise pas les fonctions précédentes.

```
1  import numpy.random as rd
2  import matplotlib.pyplot as plt
3
4  def bernoulli(p):
5      r=rd.random()
6      if r<p:
7          return 1
8      else :
9          return 0
10
11 def liste_bernoulli(n,p):
```

```python
    return [bernoulli(p) for k in range(n)]

def binomiale(n,p):
    return sum(liste_bernoulli(n,p))

def binomiale_bis(n,p):
    nb_succes=0
    for k in range(n): #ou range(0,n)
        if rd.random()<p:
            nb_succes+=1 #ou nb_succes=nb_succes+1
    return nb_succes
```

I.2 Loi géométrique

4. En utilisant la fonction **bernoulli**, écrire une fonction **geometrique** prenant en argument d'entrée un réel p de $]0;1[$ et renvoyant une réalisation d'une variable aléatoire suivant la loi $\mathscr{G}(p)$.
5. Proposer une autre fonction permettant de simuler une réalisation d'une variable aléatoire suivant la loi $\mathscr{G}(p)$ qui n'utilise pas la fonction **bernoulli**.

```python
import numpy.random as rd

def bernoulli(p):
    r=rd.random()
    if r<p:
        return 1
    else :
        return 0

def geometrique(p):
    rang=1
    while bernoulli(p)==0:
        rang+=1
    return rang

def geometrique_bis(p):
    rang=1
    while rd.random()<1-p:
        rang+=1
    return rang
```

II Simuler les lois usuelles avec les commandes existantes

Des commandes à connaître :

- **rd.random(N)** renvoie N réels aléatoires de $]0;1[$ dans un tableau 1 ligne $\times$ N colonnes
- **rd.randint(a,b)** renvoie un entier aléatoire de $[\![a;b[\![$
- **rd.randint(a,b,N)** renvoie N entiers aléatoires de $[\![a;b[\![$ dans un tableau 1 ligne $\times$ N colonnes
- **rd.binomial(n,p)** renvoie une réalisation d'une variable aléatoire suivant la loi $\mathscr{B}(n;p)$
- **rd.binomial(n,p,N)** renvoie N réalisations d'une variable aléatoire suivant la loi $\mathscr{B}(n;p)$ dans un tableau 1 ligne $\times$ N colonnes
- **rd.geometric(p)** renvoie une réalisation d'une variable aléatoire suivant la loi $\mathscr{G}(p)$
- **rd.geometric(p,N)** renvoie N réalisations d'une variable aléatoire suivant la loi $\mathscr{G}(p)$ dans un tableau 1 ligne $\times$ N colonnes
- **rd.poisson(lam)** renvoie une réalisation d'une variable aléatoire suivant la loi $\mathscr{P}(\lambda)$, avec $\lambda =$**lam**
- **rd.poisson(lam,N)** renvoie N réalisations d'une variable aléatoire suivant la loi $\mathscr{P}(\lambda)$, avec $\lambda =$**lam**, dans un tableau 1 ligne $\times$ N colonnes

✘ Attention !
Comme pour la commande **range(a,b)**, la borne de droite est exclue !

6. 6.a. Représenter l'histogramme des fréquences de 10000 réalisations indépendantes d'une variable aléatoire suivant la loi binomiale de paramètres $n = 10$ et $p = 0,3$.
   6.b. Sur le même graphique, faire apparaître les valeurs des probabilités d'une telle loi.

Aide
Après avoir importé la bibliothèque **math**, la commande **math.comb(n,k)** renvoie la valeur de $\binom{n}{k}$.

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

n=15
p=0.05
Labs=[-0.5+k for k in range(0,n+2)]
plt.hist(rd.binomial(n,p,10000),Labs,density=True,edgecolor='b')

Lproba=[math.comb(n,k)*p**k*(1-p)**(n-k) for k in range(0,n+1)]
plt.plot(range(0,n+1),Lproba,"r+")

plt.show()
```

Petite remarque

Le fait que la distribution empirique observée sur un échantillon de grande taille se rapproche de la distribution théorique est assez naturel. Nous le justifierons mathématiquement dans le chapitre 11 !

7. 7.a. Représenter l'histogramme des fréquences de 10000 réalisations indépendantes d'une variable aléatoire suivant la loi géométrique de paramètre $p = 0,5$.
   7.b. Sur le même graphique, faire apparaître les valeurs des probabilités d'une telle loi.

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

p=0.5
L=rd.geometric(p,10000)
m=max(L)
Labs=[-0.5+k for k in range(1,m+2)]
plt.hist(L,Labs,density=True,edgecolor='b')

Lproba=[p*(1-p)**(k-1) for k in range(1,m+1)]
plt.plot(range(1,m+1),Lproba,"r+")

plt.show()
```

8. 8.a. Représenter l'histogramme des fréquences de 10000 réalisations indépendantes d'une variable aléatoire suivant la loi de Poisson de paramètre $\lambda = 3$.
   8.b. Sur le même graphique, faire apparaître les valeurs des probabilités d'une telle loi.
   8.c. Comparer le graphique obtenu avec celui de la question 6....

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

lam=10
L=rd.poisson(lam,10000)
m=max(L)
Labs=[-0.5+k for k in range(0,m+2)]
plt.hist(L,Labs,density=True,edgecolor='b')

#Lproba=[lam**k*np.exp(-lam)/math.factorial(k) for k in range(0,m+1)]
#plt.plot(range(0,m+1),Lproba,"r+")

plt.show()
```

III Quelques autres lois...

III.1 Géométrique tronquée

On lance successivement et de façon indépendante une pièce donnant PILE avec la probabilité 0,3 jusqu'à obtenir le premier PILE, et on s'arrête dans tous les cas après 10 lancers. On note X la variable aléatoire égale au nombre de lancers effectués.

9. 9.a. Écrire une fonction telle que l'exécution de `simul_X()` renvoie une réalisation de la variable aléatoire X.
   9.b. Représenter l'histogramme des fréquences de 10000 réalisations indépendantes de X.

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

def simul_X():
    n=1
    while n<10:
        if rd.random()<0.7:
            n=n+1
        else :
            return n
    return n

def simul_Xbis():
    n=1
    while n<10 and rd.random()<0.7:
        n=n+1
    return n

def simul_Xter():
    for n in range(1,11):
        if rd.random()<0.3:
            return n
    return 10

L=[simul_Xter() for k in range(10000)]
Labs=[-0.5+k for k in range(1,12)]
plt.hist(L,Labs,density=True,edgecolor='k')
plt.show()
```

9.c. Donner une valeur approchée de $\mathbb{P}([X=10])$ et calculer sa valeur exacte.

- D'après l'histogramme obtenu, on peut penser que $\mathbb{P}([X=10]) \simeq 0,4$.
- Calculons cette probabilité.
  - ⋆ Notons, pour tout $k \in [\![1;10]\!]$, P_k l'évènement "obtenir PILE au lancer k" et $F_k = \overline{P_k}$.

L'évènement $[X=10]$ est réalisé $\overset{\star}{\iff}$ si, et seulement si, on a effectué 10 lancers
si, et seulement si, le premier PILE a été obtenu au dixième lancer ou aucun PILE n'a été obtenu lors des 10 lancers

D'où :

$$[X=10] = \left(\left(\bigcap_{k=1}^{9} F_k\right) \cap P_{10}\right) \cup \left(\bigcap_{k=1}^{10} F_k\right)$$

Ainsi :

$$\begin{aligned}
\mathbb{P}([X=10]) &= \mathbb{P}\left(\left(\left(\bigcap_{k=1}^{9} F_k\right) \cap P_{10}\right) \cup \left(\bigcap_{k=1}^{1} 0F_k\right)\right) \\
&= \mathbb{P}\left(\left(\bigcap_{k=1}^{9} F_k\right) \cap P_{10}\right) + \mathbb{P}\left(\bigcap_{k=1}^{1} 0F_k\right) \\
&= 0,7^9 \times 0,3 + 0,7^{10} \\
&= 0,7^9(0,3+0,7) \\
&= 0,7^9
\end{aligned}$$

(F_{10} et P_{10} sont incompatibles, donc $\left(\left(\bigcap_{k=1}^{9} F_k\right) \cap P_{10}\right)$ et $\left(\bigcap_{k=1}^{1} 0F_k\right)$ également)

(indépendance des lancers)

Petite remarque

Ici, on peut remarquer que $\left(\left(\bigcap_{k=1}^{9} F_k\right) \cap P_{10}\right) \cup \left(\bigcap_{k=1}^{1} 0F_k\right) = \left(\bigcap_{k=1}^{9} F_k\right) \cap (P_{10} \cup F_{10}) = \bigcap_{k=1}^{9} F_k$...

Petite remarque

On pourrait raisonner autrement... $[X=10]$ est réalisé si, et seulement si, on a effectué 10 lancers ; si, et seulement si, on ne s'arrête pas avant le dixième lancer ; si, et seulement si, les lancers 1 à 9 ont tous donné FACE.

D'où : $[X=10] = \bigcap_{k=1}^{9} F_k$...

III.2 Maximum de deux VA

Soit $n \in [\![2;+\infty[\![$. On considère une urne composée de n boules, numérotées de 1 à n, indiscernables au toucher. On tire simultanément deux boules dans cette urne. On note X_n la variable aléatoire égale au plus grand des deux nombres obtenus.

10. 10.a. Écrire une fonction telle que l'exécution de `simul_X(n)` renvoie une réalisation de la variable aléatoire X_n.
    10.b. Écrire un programme permettant d'obtenir une valeur approchée de $\mathbb{E}(X_5)$.

Petite remarque

L'interprétation de la moyenne empirique comme valeur approchée de l'espérance est basée sur la loi faible des grands nombres, que nous verrons dans le chapitre 11.

```python
import numpy.random as rd
def simul_X(n):
    a=rd.randint(1,n+1)
    b=rd.randint(1,n+1)
    while b==a:
```

```
        b=rd.randint(1,n+1)
    return max(a,b)

n=5
L=[simul_X(n) for k in range(10000)]
E=sum(L)/len(L)
print(E)
```

Petite remarque

Pour rappel, nous avons étudié la variable aléatoire X_n dans Chapitre 2 - Exercice 2.

III.3 Loi uniforme ?

Soit $n \in [\![3;+\infty[\![$. On dispose d'une urne contenant $(n-1)$ balles blanches et une balle noire. On effectue des tirages sans remise dans l'urne jusqu'à l'obtention de la balle noire. On note X_n la variable aléatoire égale au rang d'apparition de la balle noire.

11. 11.a. Écrire une fonction `Python` telle que l'exécution de la commande `simul_X(n)` renvoie une réalisation de la variable aléatoire X_n.

    11.b. Représenter l'histogramme des fréquences de 10000 réalisations indépendantes de X_n dans les cas $n=5$, puis $n=10$.
    Que peut-on conjecturer ?

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

def simul_X(n):
    nb=n #nb de balles restantes
    X=1
    while rd.randint(1,nb+1)!=1: #on code par 1 la balle noire
        nb=nb-1
        X=X+1
    return X

def simul_Xbis(n):
    nb=n #nb de balles restantes
    X=1
    while rd.random()<1-1/nb: #ou rd.random()<(nb-1)/nb
        nb=nb-1
        X=X+1
    return X

def simul_Xter(n):
    L=['N']+['B' for k in range(n-1)]
    i=rd.randint(len(L))
    X=1
    while L[i]=='B':
        del(L[i])
        X=X+1
        i=rd.randint(len(L))
    return X

n=10
L=[simul_Xter(n) for k in range(100000)]
Labs=[-0.5+k for k in range(1,n+2)]
plt.hist(L,Labs,edgecolor='k',density=True)
plt.show()
```

Après visualisation de l'histogramme, on conjecture que X_n suit la loi uniforme sur $[\![1;n]\!]$.

11.c. Démontrer cette conjecture.

Notons, pour tout $i \in [\![1;n]\!]$: B_i l'évènement "tirer une balle blanche au tirage i" et $N_i = \overline{B_i}$.

- Ensuite, on a déjà $X_n(\Omega) = [\![1;n]\!]$.
- Puis, soit $k \in [\![1;n]\!]$. On a :

$$[X_n = k] = \left(\bigcap_{i=1}^{k-1} B_i\right) \cap N_k$$

Petite remarque

Avec comme convention $\bigcap_{i\in\varnothing} B_i = \Omega$ (dans le cas où $k=1$).

D'où :

$$\mathbb{P}([X_n=k]) = \mathbb{P}\left(\left(\bigcap_{i=1}^{k-1} B_i\right) \cap N_k\right)$$

↙ formule des probabilités composées, licite car $\mathbb{P}(B_1 \cap \ldots \cap B_{k-1}) \neq 0$

$$= \mathbb{P}(B_1) \times \ldots \times \mathbb{P}_{B_1 \cap \ldots \cap B_{k-2}}(B_{k-1}) \times \mathbb{P}_{B_1 \cap \ldots \cap B_{k-1}}(N_k)$$

Soit ensuite $i \in [\![1; k-1]\!]$. Supposons l'évènement $B_1 \cap \ldots \cap B_i$ réalisé. On a ainsi tiré i balles blanches dans l'urne.
Le prochain tirage s'effectue donc dans une urne composée de $n-i$ balles, dont 1 est noire et $n-i-1$ sont blanches.
Par équiprobabilité du choix de la balle, on a alors :

$$\mathbb{P}_{B_1 \cap \ldots \cap B_i}(B_{i+1}) = \frac{n-i-1}{n-i}$$

$$\mathbb{P}_{B_1 \cap \ldots \cap B_i}(N_{i+1}) = \frac{1}{n-i}$$

D'où, en reprenant le calcul ci-dessus :

$$\begin{aligned}
\mathbb{P}([X_n = k]) &= \frac{n-1}{n} \times \frac{n-2}{n-1} \times \ldots \times \frac{n-1-(k-2)}{n-(k-2)} \times \frac{1}{n-(k-1)} \\
&= \frac{n-1}{n} \times \frac{n-2}{n-1} \times \ldots \times \frac{n-k+1}{n-k+2} \times \frac{1}{n-k+1} \qquad \text{télescopage} \\
&= \frac{1}{n}
\end{aligned}$$

Petite remarque

Cette partie n'est pas absolument nécessaire (si elle n'est pas demandée !) et peut être passée en première lecture...

♡ Astuce du chef ! ♡

En ne simplifiant pas immédiatement les expressions $\frac{n-1-(k-2)}{n-(k-2)}$, on comprend mieux d'où viennent ses quantités : le quotient du nombre de balles blanches restantes après en avoir tiré $k-2$, par le nombre de balles restantes dans l'urne...

Conclusion : $X_n \hookrightarrow \mathscr{U}([\![1; n]\!])$.