Chapitre 9
Simulation d'expériences aléatoires

1 Commandes `rand` et `randint` 1

2 Simulations de variables aléatoires 1
2.1 Loi de Bernoulli 1
2.2 Loi binomiale 2
2.3 Simulation d'une variable aléatoire finie de loi donnée 2

3 Estimation de probabilités et d'espérances 3
3.1 Estimation de la probabilité d'un évènement 3
3.2 Estimation de l'espérance d'une variable aléatoire 3

1 Commandes `rand` et `randint`

Pour simuler informatiquement des expériences aléatoires, nous allons utiliser les fonctions `rand` et `randint` du sous-module `random` de la bibliothèque `numpy` :

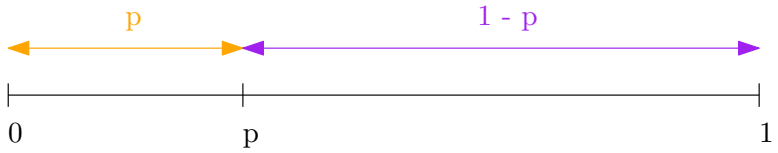
```
import numpy.random as nr
```

- L'instruction `nr.rand()` renvoie un flottant aléatoire de $[0;1[$, tiré selon la loi uniforme.
- Pour tous entiers a et b, l'instruction `nr.randint(a, b)` renvoie un entier aléatoire de $[\![a;b[\![$, tiré selon la loi uniforme sur $[\![a;b-1]\!]$.

2 Simulations de variables aléatoires

2.1 Loi de Bernoulli

Commençons par remarquer que si l'on tire un nombre au hasard (uniformément) dans l'intervalle $[0;1[$ et si $p \in [0;1]$, alors le nombre aura une probabilité de p de se trouver dans l'intervalle $[0;p[$.



La fonction suivante simule alors une variable aléatoire qui suit une loi de Bernoulli.

```
def Bernoulli(p):
    """
    Entrée : un flottant p compris entre 0 et 1.
    Sortie : une valeur qui suit une loi de Bernoulli de paramètre p.
    """
    u = nr.rand()
    if u < p: # L'évènement (u < p) se réalise avec probabilité p.
        return 1
    else:
        return 0
```

2.2 Loi binomiale

Pour simuler une variable aléatoire suivant une loi binomiale de paramètres n et p, on somme les résultats de n simulations indépendantes de variables de Bernoulli de paramètre p.

```
def binomiale(n, p):
    """
    Entrées :
        - un entier naturel n ;
        - un flottant p compris entre 0 et 1.
    Sortie : une valeur qui suit une loi binomiale
             de paramètres n et p.
    """
    somme = 0
    for epreuve in range(n):
    # On réalise n épreuves de Bernoulli indépendantes.
        somme += Bernoulli(p)
    return somme
```

2.3 Simulation d'une variable aléatoire finie de loi donnée

Admettons le résultat suivant.

Théorème. Soit n un entier naturel non nul.
Soit $(x_i)_{i\in[\![1;n]\!]}$ une famille de n réels distincts.

Soit $(p_i)_{i\in[\![1;n]\!]}$ une famille de n réels de $[0;1]$ tels que $\sum_{i=1}^{n} p_i = 1$.

Alors il existe une variable aléatoire X définie sur un univers fini Ω telle que

$$X(\Omega) = \{x_i \,|\, i \in [\![1;n]\!]\} \text{ et } \forall i \in [\![1;n]\!], \mathbb{P}(X = x_i) = p_i.$$

Pour simuler une telle variable aléatoire, on procède de la manière suivante.
On tire aléatoirement un réel u de $]0;1[$ (selon la loi uniforme sur $]0;1[$).

- ⋆ Si $u < p_1$, ce qui arrive avec probabilité p_1, alors on renvoie x_1.
- ⋆ Si $p_1 \leqslant u < p_1 + p_2$, ce qui arrive avec probabilité p_2, alors on renvoie x_2.
- ⋆ Si $p_1 + p_2 \leqslant u < p_1 + p_2 + p_3$, ce qui arrive avec probabilité p_3, alors on renvoie x_3.
- ⋆ Et ainsi de suite.

```
def simulation(V, P):
    """
    Entrées :
        - la liste V des valeurs prises par la variable aléatoire ;
        - la liste P des probas que la VA tombe sur ces valeurs.
    Sortie : la valeur de la valeur aléatoire sur une issue.
    """
    u = nr.rand()
    somme = 0
    for k in range(len(P)): # k parcourt les indices de la liste P
                            # (qui sont aussi ceux de la liste V).
        somme += P[k]
        if u < somme:
            return V[k]
```

```
[simulation([-2, 0, 14, 5], [0.3, 0.2, 0.4, 0.1]) for s in range(1000)]
```

3 Estimation de probabilités et d'espérances

3.1 Estimation de la probabilité d'un évènement

Considérons une expérience aléatoire et un évènement dont on souhaite estimer la probabilité p.

Si l'on répète N fois une expérience aléatoire, alors l'évènement se réalisera en moyenne dans pN des expériences.

Ainsi, pour estimer la probabilité p de cet évènement, on répète un *grand* nombre N de fois l'expérience et on compte le nombre C de fois où l'évènement s'est réalisé. La probabilité p de l'évènement sera environ égale à la fréquence $\dfrac{C}{N}$ de réalisation de l'évènement sur les N expériences.

3.2 Estimation de l'espérance d'une variable aléatoire

La loi des grands nombres, qui est au programme de seconde année, stipule que sur un nombre de simulations indépendantes d'une variable aléatoire suffisamment grand, la moyenne des résultats obtenus fournit une bonne approximation de l'espérance de la variable aléatoire.

Estimons alors numériquement l'espérance d'une loi binomiale.

```
def moyenne_empirique_binomiale(n, p, N):
    """
    Entrées :
        - un entier naturel n ;
        - un flottant p compris entre 0 et 1 ;
        - un entier N (grand).
    Sortie : moyenne empirique d'une loi binomiale
             de paramètres n et p
             calculée sur N simulations indépendantes.
    """
    S = 0
    for simulation in range(N): # On réalise N simulations.
        S += binomiale(n, p)
    moyenne = S / N
    return moyenne
```

Testons alors cette fonction avec différentes valeurs de N.

```
moyenne_empirique_binomiale(10, 0.3, 1000)
```

Nous verrons en TP comment, sur le même principe, estimer numériquement la variance d'une variable aléatoire.