INF 6 - Simulation de variables aléatoires finies.

Le langage Python permet aussi de faire des études de probabilités. Dans ce premier chapitre nous allons voir comment simuler des variables aléatoires réelles finies.

La clé sera de savoir simuler des lois usuelles, c'est à partir d'elles qu'on pourra simuler l'ensemble des lois.

On commencera toujours par importer la librairie pythonnumpy.random avec la commande suivante.

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

Ainsi les méthodes de `numpy.random` seront appelées à l'aide de `rd.xxxxx`. Dans ce cours, on se limitera aux méthodes de simulation des variables aléatoires finies connues.

1. SIMULATION DES LOIS USUELLES

Simuler une loi uniforme. On simule une loi uniforme sur des entiers avec `rd.randint`. Il y a différentes façons de l'utiliser.

1. `rd.randint(n)` renvoie un entier aléatoire uniforme entre 0 et $n-1$.
2. `rd.randint(a,b)` renvoie un entier aléatoire uniforme entre a et $b-1$.
3. on peut rajouter un argument `size = n` ou `size = m, n` pour générer un certain nombre de nombres aléatoires pour remplir un tableau.
   Par exemple `rd.randint(10,size=100)` renvoie un vecteur de 100 entrées aléatoires uniformes comprises entre 0 et 9.

Simulation de Bernoulli et Binomiales Ces deux lois se simulent de la même façon, avec la commande `rd.binomial`. Ici c'est très simple, la commande

```
rd.binomial(n,p)
```

simule une VA binomiale de paramètres $n \in \mathbb{N}^*$ et $p \in [0,1]$. Pour simuler une Bernoulli, on prend $n = 1$.

De la même façon que pour les uniformes, on a une commande optionnelle `size = n` ou `size = n,m` pour simuler plusieurs variables à la fois dans un vecteur ou un tableau.

2. SIMULATION "À LA MAIN" AVEC DES NOMBRES ALÉATOIRES

Certaines expériences aléatoires sont plus faciles à simuler sans passer par les lois usuelles. Pour cela on utilise la commande
`rd.random()`

qui génère un nombre aléatoire compris entre 0 et 1. Cette fonction ne prend pas d'argument mais peut tout de même prendre l'argument optionnel `size =` qui fonctionne comme précédemment.

Exemple 1 — Que simule cette fonction ?

```
def simu(p):
    if rd.random() > p :
        return(0)
    else :
        return(1)
```

Plus tard, la notion **fonction de répartition** nous montrera qu'on peut simuler toute VA finie uniquement avec cette commande.

EXERCICES

Exercice 1 Écrire une fonction Python qui prend en argument deux entiers naturels n et p strictement positifs qui simule p tirages successifs et avec remise dans une urne contenant n boules numérotées de 1 à n et qui renvoie un vecteur contenant les p numéros tirés (dans l'ordre du tirage).

Exercice 2 Écrire une fonction qui prend en argument $n \in \mathbb{N}^*, r \in \mathbb{N}^* et p \in]0,1[$ et qui renvoie un vecteur à r coordonnées contenant des réalisations indépendantes de variables aléatoires de loi $\mathrm{B}(n,p)$ en utilisant

1. rd.binomial
2. rd.random

Exercice 3 Écrire une fonction Python qui prend en argument un entier naturel n non nul et qui renvoie le nombre de succès lorsqu'on effectue n expériences de Bernoulli indépendantes telles que, pour tout $k \in \{1, \dots n\}$, la k-ième expérience a une probabilité de succès de $\frac{1}{k+1}$.

Exercice 4 Écrire un programme qui prend en entrée un entier n et simule l'expérience suivante :

1. on a une urne avec des boules numérotées de 1 à n
2. on dire une boule : si on tire la boule 1 on s'arrête, sinon on retire toutes les boules comprises entre $k+1$ et n (inclus)
3. on répète l'expérience.

La fonction doit renvoyer le nombre de tirages fait pour obtenir la boule 1.

Réaliser l'expérience 100 fois. Calculer la moyenne des résultats obtenus.

Exercice 5 Écrire une fonction qui prend en entrée un entier n et réalise l'expérience suivante :

1. On simule n variables de Rademacher $\mathrm{X}_1, \dots, \mathrm{X}_n$.
2. On calcule les termes de la suite

$$\mathrm{S}_k = \sum_{i=1}^{n} \mathrm{X}_k.$$

On tracera la suite S_k. On pourra importer la bibliothèque `matlotlib.pyplot` importée sous le nom `plt`.