INF 11 - Simulation d'une loi avec la fonction rand.

Dans ce TP, on va voir comment utiliser la fonction `rd.random()` pour simuler deux types de variables aléatoires.

On rappelle que la fonction `rd.random()` renvoie un nombre aléatoire uniformément dans $[0,1]$.

Simulation d'une variable de Bernoulli Pour simuler une variable aléatoire de Bernoulli de paramètre $p \in [0,1]$, on remarque que si on simule un nombre aléatoire entre 0 et 1, alors la probabilité qu'il soit dans $[0,p]$ vaut p, et donc vaut $P(X=1)$. Cela fournit le code suivant

```
import numpy.random as rd

def simu_bernoulli(p):
    x = rd.random()
    if x < p :
        return(1)
    else:
        return(0)
```

Simulation d'une variable binomiale Pour simuler une loi binomiale de paramètres n et p on a plusieurs choix. Le premier est naturel et laissé en exercice.

Exercice 1 A partir de la fonction précédente, écrire une fonction qui prend en entrée un entier n et un réel $p \in [0,1]$ et simule une loi binomiale $B(n,p)$.

L'autre solution est l'utilisation de la **fonction de répartition**. On définit la fonction de répartition d'une variable aléatoire discrète X par

$$\forall n \in \mathbb{N}, F_X(n) = P(X \leqslant n).$$

On peut alors construire le "tableau" des valeurs de la fonction de répartition d'une variable aléatoire $B(n,p)$. Pour cela, on utilise le fait que

$$\begin{aligned}\forall k \in \{0, n-1\}, F_X(k+1) &= F_X(k) + P(X=n)\\ &= F_X(k) + \binom{n}{k+1} p^{k+1}(1-p)^{n-k-1}\\ &= F_X(k) + \frac{p}{1-p}\frac{n-k}{k+1}\binom{n}{k} p^k (1-p)^{n-k}\\ &= F_X(k) + \frac{p}{1-p}\frac{n-k}{k+1} P(X=k)\end{aligned}$$

ce qui permet de calculer de proche en proche les probabilités et donc la fonction de répartition.

```
def fonction_repartition_binomiale(n,p):
    F = np.zeros(n+1) #on crée une tableau à n+1 entrées
    P = (1-p)**n
    F[0] = P # on initialise le calcule de la probabilité
    for i in range(1,n+1):
        P = (p/(1-p))*((n-i+1)/(i))*P #mise a jour de la probabilité
        F[i]=F[i-1] + P #remplissage du tableau
    return(F)
```

Une fois qu'on a crée le tableau on a juste à simuler une variable uniforme entre 0 et 1, et parcourir le tableau jusqu'à trouver une valeur qui lui est supérieure. A ce moment là, on s'arrête et on renvoie le résultat d'avant.

Exercice 2 A l'aide de la fonction et de la remarque précédente, coder un autre programme qui simule une variable binomiale.

Simulation d'une variable géométrique Nous n'avons pas encore vu la loi géométrique en classe.

Définition 1 | Variable aléatoire géométrique

Une variable aléatoire X suit une loi géométrique de paramètre $p \in]0,1[$ si :
- $X(\Omega) = \mathbb{N}^*$,
- $\forall n \in \mathbb{N}^*, P(X=n) = p(1-p)^{n-1}$.

La loi géométrique est en fait la loi du premier succès. Si on réalise une série d'épreuves de Bernoulli successives et indépendantes, la variable aléatoire est celle qui compte combien on a dû réaliser d'expériences pour avoir un premier succès.

Nous allons voir deux façons de simuler une variable aléatoire géométrique de paramètre p. La première est de simuler autant de loi Bernoulli de paramètre p jusqu'à obtenir une réalisation qui renvoie 1. On compte le nombre de simulations faites et on renvoie ce nombre là en sortie.

Exercice 3 Coder une fonction qui prend en entrée un paramètre $p \in [0,1]$ et renvoie la réalisation d'une loi géométrique de ce paramètre. On utilisera la construction donnée ci-dessus.

On peut aussi réaliser la simulation avec un seul nombre aléatoire en utilisant le même principe que pour la binomiale (avec la fonction de répartition). Mais cette fois, on ne pourra pas fixer la taille d'un tableau car on ne sait pas qu'on on s'arrêtera! La mise à jour de la probabilité se fera par l'initialisation $F_X(1) = p$ et

$$\forall n \in \mathbb{N}, F_X(n+1) = F_X(n) + P(X = n+1) = F_X(n) + p(1-p)^n = F_X(n) + (1-p)P(X = n).$$

```
def simu_geometrique2(p):
    F = p
    P = p
    n = 1
    x = rd.random()
    while x > F:
        P= (1-p)*P
        F = F + P
        n = n+1
    return(n)
```

Prenez quelques minutes pour bien comprendre ce que fait ce programme!