TP 16 : variables aléatoires discrètes usuelles

Exercice 1.

On effectue une infinité de lancers successifs d'un dé équilibré, lancers qu'on suppose indépendants.
On note X_1, la variable aléatoire égale au nombre de lancers effectués jusqu'à l'obtention du premier 6.
On note X_2 la variable aléatoire égale au nombre de lancers effectués jusqu'à l'obtention du deuxième 6.
On note de même X_3, X_4, etc...

1. *Première approche*
   (a) À l'aide d'une boucle `while`, écrire une fonction `simulX1()` qui renvoie une réalisation de X_1
   (b) Écrire de même une fonction Python qui simule X_2.
   (c) Écrire une fonction `simulX(n)` qui simule X_n, où n est donné par l'utilisateur.
2. *Deuxième approche*
   On admet que si $n \leqslant 1000$, la probabilité de $[X_n \leqslant 10000]$ est presque 1.
   (a) Créez un vecteur numpy X contenant 10000 entiers choisis uniformément entre 1 et 6.
   (b) À l'aide de l'instruction `np.where(X==6)` dont vous découvrirez le comportement vous-mêmes, obtenez la liste des positions des 6 dans le vecteur X.
   (c) En déduire une fonction `simulX_v2(n)` qui simule X_n.
3. *Finalement...*
   (a) Reconnaître la loi de X_1. Reconnaître la loi de $X_2 - X_1$.
   (b) En déduire une autre version `simulX_v3(n)` de votre programme.
   (c) Vérifier vos prédictions à l'aide de diagrammes en barres.

Exercice 2.

On lance une pièce ayant probabilité p de faire pile et on note Z la variable aléatoire égale au rang du lancer où l'on obtient le premier pile.

Après cette série de lancers, si Z a pris la valeur k ($k \in \mathbb{N}^*$), on remplit une urne de k boules numérotées $1, 2, \ldots, k$, puis on extrait au hasard une boule de cette urne.

On note X la variable aléatoire égale au numéro de la boule tirée après la procédure décrite ci-dessus.

1. Reconnaître la loi de Z.
2. Sachant $[Z = k]$, avec $k \in \mathbb{N}^*$, quelle serait alors la loi de X ?
3. En déduire une fonction `simulation(p)` qui prend en argument la valeur de p et qui renvoie une réalisation de la variable X.
4. Tracer son histogramme et donner une estimation numérique de son espérance.

Exercice 3 (Ecricome 2019)**.**

Soit D une variable aléatoire prenant les valeurs -1 et 1 avec équiprobabilité.

Écrire une fonction `D(n)` de **trois lignes en tout**, qui prend un entier n en entrée, et renvoie un vecteur numpy contenant n réalisations de la variable aléatoire D.

Exercice 4 (EDHEC 2018).

On dispose de trois pièces : une pièce numérotée 0*, pour laquelle la probabilité d'obtenir "pile" vaut* $1/2$ *et celle d'obtenir "face" vaut également* $1/2$*, une deuxième pièce numérotée* 1*, donnant "face" à coup sûr et une troisième pièce, numérotée* 2*, donnant "pile" à coup sûr.*

On choisit l'une de ces pièces au hasard et on la lance indéfiniment.

On considère la variable aléatoire X*, égale au rang d'apparition du premier "pile" et la variable aléatoire* Y*, égale au rang d'apparition du premier "face". On convient de donner à* X *la valeur* 0 *si l'on n'obtient jamais "pile" et de donner à* Y *la valeur* 0 *si l'on n'obtient jamais "face".*

On décide de coder "pile" par 1 *et "face" par* 0 *.*

1. Compléter le script `Python` suivant pour qu'il permette le calcul et l'affichage de la valeur prise par la variable aléatoire X lors de l'expérience réalisée dans cet exercice.

```
piece = rd.randint( .... )
X = 1
if piece == 0:
    lancer = rd.randint( .... )
    while lancer == 0:
        lancer = ....
        X = ....
else:
    if piece == 1:
        X = ....
print(X)
```

2. Pourquoi le cas `piece == 2` ne semble t-il pas pris en compte dans le script précédent ?

Exercice 5 (EML 2018).

Dans cette exercice, p *désigne un réel de* $]0;1[$.
Deux individus A *et* B *s'affrontent dans un jeu de Pile ou Face dont les règles sont les suivantes :*
• *Le joueur* A *dispose d'une pièce faisant Pile avec probabilité* $2/3$ *et lance cette pièce jusqu'à l'obtention du deuxième Pile ; on note* X *la variable aléatoire prenant la valeur du nombre de Face obtenus.*
• *Le joueur* B *dispose d'une autre pièce amenant Pile avec la probabilité* p *et lance cette pièce jusqu'à l'obtention d'un Pile ; on note* Y *la variable aléatoire prenant la valeur du nombre de Face alors obtenus.*
• *Le joueur* A *gagne si son nombre de Face obtenus est inférieur ou égal à celui de* B *; sinon c'est le joueur* B *qui gagne.*
On dit que le jeu est équilibré lorsque les joueurs A *et* B *ont la même probabilité de gagner.*

1. Écrire une fonction `Python` `simX()` qui simule la variable aléatoire X.
2. On suppose que l'on dispose d'une fonction `simY` qui, prenant en argument un réel p de $]0;1[$, simule la variable aléatoire Y. Expliquer ce que renvoie alors la fonction suivante :

```
def mystere(p):
    r = 0
    N = 10**4
    for k in range(N):
        x = simX()
        y = simY(p)
        if x <= y:
            r = r + 1
    return r/N
```

3. Codez `simY`, puis créez un vecteur `numpy` contenant 100 valeurs de p, réparties entre 0.01 et 1.
   Tracer en fonction de p, une estimation de la probabilité que A gagne.
   À la vue de ce graphe, conjecturer une valeur de p pour lequel le jeu serait équilibré.

Formulaire :

Vous aurez besoin des commandes suivantes (dans le désordre)

```
from math import *
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as rd
rd.randint(...)
rd.random(...)
rd.binomial(...)
rd.geometric(...)
rd.poisson(...)
np.linspace
for ... in ... :
def ... ( ... ) :
return
print( ... )
if ... :
else :
log
[ ... for ... in ...]
plt.plot( ... , ... )
plt.bar(...)
```