TP # 15

Variables aléatoires en Python (Part. 1)

Pour manipuler des instructions ayant trait à l'aléatoire, on commencera toujours par importer la bibliothèque `numpy.random`. L'importation privilégiée est la suivante :

Bibliothèque numpy.random

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

Instructions aléatoires en Python

La commande `rd.random()` génère un nombre réel "aléatoire" choisi dans l'intervalle $[0,1]$.
Ceci correspond à une réalisation d'une variable aléatoire U suivant
la loi uniforme sur $[0,1]$: $U \hookrightarrow \mathcal{U}([0,1])$ (loi étudiée en détail l'année prochaine...)

rd.random

- `rd.random()` génère un nombre réel uniformément choisi dans $[0,1]$.
- `rd.random(n)` génère un vecteur ligne de taille $1 \times n$
  contenant de tels nombres aléatoires générés indépendamment.
- `rd.random( (n,p))` génère un tableau de taille $n \times p$
  contenant de tels nombres aléatoires générés indépendamment.

Exercice 1

Après avoir importé la bibliothèque `numpy.random` :

1. Tester la commande `rd.random()` dans la console plusieurs fois d'affilée.

Quelques résultats obtenus : .....

.....

.....

2. Tester la commande `rd.random(5)` dans la console plusieurs fois d'affilée.

On admet qu'une variable $U \hookrightarrow \mathcal{U}([0,1])$ satisfait la propriété suivante :

$$\forall p \in [0,1], \quad P(U \leqslant p) = P(U < p) = p.$$

Par exemple : • L'évènement $[U \leqslant 1]$ est réalisé avec probabilité 1 (logique !)

• $[U \leqslant \frac{1}{2}]$ est réalisé avec probabilité $\frac{1}{2}$, $[U \leqslant \frac{1}{3}]$ avec probabilité $\frac{1}{3}$ etc...

On retiendra donc : `rd.random()` $<$ `p` $= \begin{cases} \texttt{True} & \text{avec probabilité } p \\ \texttt{False} & \text{avec probabilité } 1-p \end{cases}$

Effectuer une instruction "aléatoirement"

La syntaxe suivante permet de réaliser l'instruction 1 avec probabilité p
ou bien l'instruction 2 avec probabilité $1-p$:

```
if rd.random() < p :
        instruction1
else :
        instruction2
```

Construction de la loi de Bernoulli

Exercice 2

1. En s'inspirant de l'idée donnée dans l'encadré précédent, définir une fonction `bernoulli` qui prend en entrée un réel $p \in [0,1]$ et simule une réalisation d'une variable aléatoire de loi $\mathcal{B}(p)$ (elle devra donc renvoyer 0 ou 1).

```
import numpy.random as rd
def bernoulli(p) :
```

2. Tester l'instruction `bernoulli(1/3)` 10 fois d'affilée dans la console.

Résultats obtenus : .....

Proportion de "1" obtenus sur ces 10 réalisations : .....

3. Après un grand nombre d'appels de l'instruction `bernoulli(1/3)`, on s'attend bien-sûr à ce que la proportion de "1" obtenus se rapproche de 1/3...
Compléter le programme pour afficher cette proportion sur $N = 1000$ appels.

```
N = 1000; S = ....

for k in range(N) :

    S = S + ..........

print("Proportion de 1 :", .............)
```

Proportion de "1" sur 1000 appels : ..... Sur 10000 appels : .....

Construction de la loi binomiale

Exercice 3

Rappel : Une variable aléatoire de loi $\mathcal{B}(n,p)$ compte le nombre de succès lors de n répétitions indépendantes d'une épreuve de Bernoulli de paramètre p.

1. A partir de ce rappel, compléter la fonction `binomiale` pour qu'elle simule la réalisation d'une variable aléatoire de loi $\mathcal{B}(n,p)$.

```
def binomiale(n,p) :

    X = 0
    for k in range( ......... ) :

        if ............................ :

            X = X + 1

    return(X)
```

2. Tester l'instruction `binomiale( 10 , 0.5 )` 10 fois d'affilée dans la console.

Résultats obtenus : .....

Exercice 4

1. Tester et comprendre les instructions suivantes dans la console :

```
>>> import numpy as np

>>> A = rd.random(5)    >>> print(A)

>>> print(A < 0.5)    >>> np.sum(A < 0.5)

>>> print(A < 0.8)    >>> np.sum(A < 0.8)
```

Pour un vecteur `A` et un nombre `x`, que renvoie l'instruction `np.sum( A < x )` ?

.....

2. En déduire une façon plus succincte de simuler une loi $\mathcal{B}(n,p)$:

```
import numpy as np

def binomiale(n,p) :

    X = ..........................................

    return(X)
```

Explication :

BONUS (variantes) :

Exercice 5

On réalise une succession de n épreuves de Bernoulli indépendantes.
La probabilité de succès de l'épreuve numéro k est $1/k$.
Soit X le nombre de succès obtenus à l'issue des n épreuves.
En s'inspirant du programme de l'Exercice 3, compléter la fonction `alea` qui simule une réalisation de la variable aléatoire X.

```
def alea(n) :









    return(X)
```

Exercice 6

Un jeu vidéo est constitué d'une infinité de niveaux : niveau 1, niveau 2, etc...
Dès qu'un niveau est perdu, la partie s'arrête.
On note X le nombre de niveaux complétés par le joueur à l'issue de sa partie.

1. On suppose que la probabilité de compléter chaque niveau est $1/2$.
Compléter le script suivant pour simuler une réalisation de X :

```
X = 0

while ........................ :

    X = X + 1

print(X)
```

2. On suppose que la probabilité de compléter le niveau k est $1/k$.
Compléter le script suivant pour simuler une réalisation de X :

```
X = 0

while ........................ :

    X = X + 1

print(X)
```