TP 15 : simulation de variables aléatoires

Pour tirer informatiquement un nombre aléatoire, on utilise le sous-module Python `random`*, partie intégrante de* `numpy`*. On écrira au début de tous les scripts*

```python
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
```

Dans ce TP nous utiliserons seulement les méthodes `random` et `randint` du module `random`.
Voici quelques exemples d'exécution de `rd.randint` :

```python
>>> rd.randint(0, 4, 10)
array([3, 1, 1, 0, 1, 3, 0, 1, 1, 1])
>>> rd.randint(0, 4, 10)
array([3, 2, 2, 2, 0, 1, 0, 1, 3, 3])
>>> rd.randint(-57, 38, 1)
array([-10])
>>> rd.randint(-57, 38, 1)
array([36])
>>> rd.randint(-57, 38)
14
>>> rd.randint(-57, 38)
-24
>>> for k in range(10):
....     print(rd.randint(7), end = " ")
4 6 1 1 0 4 0 1 4 1
```

Vous l'aurez compris, `rd.randint(a, b, n)` génère un vecteur `numpy` de n entiers aléatoires tirés dans l'ensemble $[\![a;b[\![$: b n'apparaîtra donc jamais, mais $b-1$ oui. Si on omet le troisième argument, on obtient seulement un entier, pas un vecteur. Si on ne donne qu'un seul argument a, on obtient un entier aléatoire entre 0 et $a-1$ inclus.

Essayons de vérifier si l'ordinateur fait le tirage de manière uniforme ou non. Pour cela on note X la variable aléatoire donnant le réel obtenu par un tirage d'un réel via `rd.randint(0,4)`, et on va tenter de mesurer la loi de X, c'est-à-dire la probabilité de l'évènement $[X=k]$, pour k dans $[\![0;3]\!]$.

Exercice 1 (Le tirage est-il uniforme ?).

1. Créez un vecteur numpy X de taille $N=100$ contenant des entiers aléatoires dans $[\![0;3]\!]$.
2. Que donne l'exécution de l'instruction `X == 2` ? Calculez alors `np.mean(X == 2)`.
3. En déduire une estimation numérique de $\mathbb{P}(X=2)$.
4. Estimez de même $\mathbb{P}(X=0)$, $\mathbb{P}(X=1)$, $\mathbb{P}(X=3)$. La loi semble-t-elle uniforme ?
5. Refaites tous ces calculs en passant à $N=10000$.

Exercice 2 (Visualisation de loi discrète).

On voudrait faire cette même vérification mais avec `rd.randint(0, 53)`*.*
Il s'agit de systématiser nos calculs des valeurs de $\mathbb{P}(X=k)$ et de représenter cette loi avec `matplotlib`*.*

1. Créez un vecteur numpy X de taille $N=1000$ contenant des entiers aléatoires dans $[\![0;54]\!]$.
2. Créez un autre vecteur numpy nommé `loiDeX` qui contiendra les estimations des $\mathbb{P}(X=k)$ pour k allant de 0 à 53, ainsi qu'un vecteur numpy `abscisses` contenant les entiers de 0 à 53.
3. Exécutez la commande `plt.bar(abscisses,loiDeX)` suivie de `plt.show()`.
   Augmentez progressivement N. Le tirage est-il uniforme ?

Voici quelques exemples d'exécution de `rd.random` :

```
>>> rd.random()
0.3454002182044935
>>> rd.random()
0.8986430303263077
>>> rd.random(3)
array([0.74318353, 0.90540151, 0.64971556])
>>> nombres = rd.random(100000)
array([0.9653995 , 0.5195159 , 0.72038433, ..., 0.57960302, 0.2015765 ,
       0.11875637])
>>> len(nombres)
100000
>>> nombres[57]
0.9014666996034315
>>> rd.random(3, 5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "mtrand.pyx", line 334, in numpy.random.mtrand.RandomState.random
TypeError: random() takes at most 1 positional argument (2 given)
```

Vous l'aurez compris, `rd.random()` génère un réel aléatoire entre 0 et 1. Essayons de vérifier si l'ordinateur fait ceci de manière uniforme ou non. Pour cela on note X la variable aléatoire donnant le réel obtenu par un tirage d'un réel via `rd.random`, et on va tenter de mesurer la fonction de répartition F_X de cette v.a. X.

Exercice 3 (Le tirage est-il uniforme entre 0 et 1 ?).

1. Créez un vecteur numpy nommé `X` de taille $N = 100$ contenant des réels aléatoires obtenus via `rd.random`.
2. Créez un vecteur `abscisses` de taille $M = 1000$ contenant M réels allant de -0.5 à 1.5.
3. Créez un vecteur `fctRepartition` de taille $M = 1000$ contenant les des estimations de $\mathbb{P}(X \leqslant k)$ pour k appartenant à `abscisses`.
4. Représentez alors F_X à l'aide de `plt.pyplot`.
5. Augmentez progressivement N. Le tirage est-il uniforme ?

Exercice 4 (La moyenne a-t-elle un sens ?).

On considére une v.a. X de support $\mathbb{N}^*$ et dont la loi est donnée par $\mathbb{P}(X = k) = \frac{1}{k(k+1)}$ pour tout $k \in \mathbb{N}^*$. On obtient cette v.a. en réalisant l'expérience suivante : on démarre avec une urne contenant une boule blanche et une noire, et à chaque étape, si on tire une boule blanche on la remet et on ajoute une boule blanche dans l'urne. Le jeu s'arréte quand on obtient une boule noire, et on note X le numéro du tirage oé on a obtenu la boule noire.

1. Créez une fonction `attenteNoire()` n'ayant pas d'argument mais qui renvoie une valeur de X. On pourra utiliser la commande `rd.binomial(1,p)` qui renvoie 0 avec probabilité $1 - p$ et 1 avec probabilité p.
2. Créez un vecteur numpy contenant $N = 10000$ réalisations de X.
3. Calculer la moyenne des valeurs de X obtenues à l'aide de `np.mean(X)`.
4. Pour chaque valeur entiére de N entre 1 et 10000, recommencer la simulation. Tracez les différentes moyennes obtenues en fonction de N.

Formulaire :

Vous aurez besoin des commandes suivantes (dans le désordre)

```
from math import *
import numpy as np
import matplotlib.pyplot as plt
np.linspace
for ... in ... :
def ... ( ... ) :
return
print( ... )
if ... :
else :
log
[ ... for ... in ...]
plt.plot( ... , ... )
plt.show()
```