CONCOURS CENTRALE•SUPÉLEC

Oral

Python

MP, PC, PSI, TSI

Probabilités

Les fonctions d'échantillonnage et de génération de valeurs pseudo-aléatoires sont regroupées dans la bibliothèque `numpy.random`.

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

L'expression `randint(a, b)` permet de choisir un entier au hasard dans l'intervalle $[\![a, b[\![$. La fonction `randint` prend un troisième paramètre optionnel permettant d'effectuer plusieurs tirages et de renvoyer les résultat sous forme de tableau ou de matrice.

```
>>> rd.randint(1, 7)          # un lancer de dé
2
>>> rd.randint(1, 7, 20)      # 20 lancers de dé
array([5, 2, 2, 3, 1, 5, 5, 3, 6, 4, 2, 6, 6, 4, 3, 2, 4, 5, 1, 3])
>>> rd.randint(1, 7, (4, 5))  # 20 lancers de dé sous forme d'une matrice 4x5
array([[3, 6, 1, 6, 3],
       [5, 1, 6, 2, 2],
       [3, 1, 2, 2, 5],
       [5, 2, 6, 1, 4]])
```

La fonction `random` renvoie un réel compris dans l'intervalle $[0, 1[$. Si X désigne la variable aléatoire correspondant au résultat de la fonction `random`, alors pour tout a et b dans $[0, 1]$ avec $a \leqslant b$, on a $P(a \leqslant X < b) = b - a$. Cette fonction accepte un paramètres optionnel permettant de réaliser plusieurs tirages et de les renvoyer sous forme de tableau ou de matrice.

```
>>> rd.random()
0.9168092013708049
>>> rd.random(4)
array([ 0.98748897,  0.86589972,  0.53683001,  0.50687386])
>>> rd.random((2,4))
array([[ 0.78230688,  0.83803526,  0.62077457,  0.27432819],
       [ 0.66522387,  0.71258365,  0.25813448,  0.28833084]])
```

La fonction `binomial` permet de simuler une variable aléatoire suivant une loi binomiale de paramètres `n` et `p`. Elle permet donc également de simuler une variable aléatoire suivant une loi de Bernoulli de paramètres `p` en prenant simplement $n = 1$. Cette fonction prend un troisième paramètre optionnel qui correspond, comme pour les fonctions précédentes, au nombre de valeurs à obtenir.

```
>>> rd.binomial(10, 0.3, 7)
array([2, 2, 2, 2, 2, 4, 3])
>>> rd.binomial(1, 0.6, 20)
array([0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1])
```

Les fonctions `geometric` et `poisson` fonctionnement de la même manière pour les lois géométrique ou de Poisson.

```
>>> rd.geometric(0.5, 8)
array([1, 1, 3, 1, 3, 2, 5, 1])
>>> rd.poisson(4, 15)
array([5, 2, 3, 4, 6, 0, 5, 3, 1, 5, 1, 5, 9, 4, 6])
```