Concours Blanc : TP d'Informatique

PTSI A et B Lycée Eiffel

20 mai 2016

Ce TP est consacré à l'étude de quelques situations probabilistes à l'aide de Python. On rappelle que l'importation de l'ensemble des fonctions du module **random** s'effectue à l'aide de la commande : from random import *. Ce module contient notamment une fonction **randint** qui permet d'effectuer des tirages aléatoirs d'entiers : randint(a,b) retourne un entier aléatoire compris entre a et b inclus tous les deux. Il contient également une fonction random() retournant un nombre décimal aléatoire compris entre 0 et 1.

Simulations de lancers de dés.

1. Écrire une commande qui simule un lancer de dé équilibré à six faces.
2. Écrire une fonction **lancerde(n)** qui effectue n lancers de dés successifs et retourne une liste contenant le nombre de fois où on est tombé sur chaque face du dé (la liste doit donc contenir six valeurs).
3. Écrire une fonction **moyennede(n)** qui effectue n lancers de dés successifs et retourne la moyenne de tous les résultats obtenus. Tester cette fonction pour $n = 10$, puis $n = 100$ etc, jusqu'à ce que les calculs commencent à devenir trop longs. On pourra commenter les résultats obtenus.
4. Écrire une fonction **premiersix()** qui effectue des lancers de dés jusqu'à obtenir un 6 et qui affiche le nombre d'essais nécessaires.
5. Écrire un programme **geometrique(n)** effectuant n fois de suite la fonction **premiersix()** et qui retourne la liste de tous les résultats obtenus. Effectuer une simulation avec $n = 1000$ et compter (pas à la main, évidemment), le nombre de fois où on a obtenu la valeur 1 (commenter le résultat). Quelle est la valeur maximale obtenue ?
6. Modifier le programme **geometrique(n)** pour qu'il retourne désormais une liste contenant le nombre de fois que chaque valeur de p comprise entre 1 et 100 a été prise lors des n simulations.
7. À l'aide du module matplotlib.pyplot, tracer une courbe représentant, en fonction de p, le nombre de fois où la simulation précédente a pris la valeur p lorsque $n = 10000$. Commenter la courbe obtenue.

Lancers de pièces.

1. Écrire un programme **piece(p)** simulant un lancer de pièce déséquilibrée tombant sur Pile avec probabilité p (et donc sur Face avec probabilité 1-p). Par souci de simplicité, le programme pourra retourner la valeur 1 si on tombe sur Pile, et 0 si on tombe sur Face. Il est fortement conseillé d'utiliser intelligemment la fonction random, et on pourra ajouter un message d'erreur si la valeur de p proposée n'est pas comprise entre 0 et 1.
2. Écrire une fonction **suitelancers(p,n)** effectuant une série de n lancers de pièces déséquilibrées, et renvoyant le nombre de Piles obtenus. Tester ce programme avec $n = 100$ et différentes valeurs de p. Commenter le résultat.

3. Effectuer 1000 fois de suite le programme **suitelancers(0.5,5)** et retranscrire dans une liste le nombre de fois où on a obtenu zéro Pile, un Pile, deux Piles, ..., cinq Piles. On pourra comenter les résultats obtenus.
4. Effectuer 10 000 fois de suite le programme **suitelancers(0.5,20)** et représenter avec matplotlib.pyplot le nombre de fois où on a obtenu chaque quantité de Pile possible entre 0 et 20. Commenter.

Marches aléatoires.

On se propose d'étudier un phénomène de marche aléatoire sur une droite : un objet se trouve initialement à l'emplacement 0 et, à chaque étape, il a une chance sur deux d'avancer d'une unité et une chance sur deux de reculer d'une unité (la droite est supposée infinie dans les deux directions).

1. Écrire une fonction **marche(n)** simulant n pas de marche aléatoire puis affichant la position finale. Modifier le programme pour qu'il affiche la liste de toutes les positions intermédiaires.
2. Modifier le programme précédent pour qu'il calcule le nombre de fois où on est repassé par 0 au cours de la marche aléatoire, ainsi que la plus grande valeur atteinte. Tester pour différentes valeurs de n (on pourra essayer de regrouper les résultats dans un tableau).
3. Calculer la moyenne de toutes les valeurs obtenues pour une marche aléatoire de 1000 étapes. Recommencer avec 10000 ou 100000, et commenter les résultats obtenus.
4. On cherche désormais à simuler une marche aléatoire en deux dimensions : l'objet se trouve initialement à la position $(0, 0)$ et peut avancer dans une des quatre directions (haut, bas, gauche et droite) avec probabilité $\frac{1}{4}$ à chaque étape. Écrire un programme Python simulant une marche aléatoire de n étapes en deux dimensions.
5. Tracer la trajectoire obtenue pour une telle marche aléatoire à l'aide de matplotlib.pyplot. On pourra tester pour différentes valeurs de n pour admirer les résultats obtenus.
6. Modifier le programme précédent pour qu'il effectue la marche non plus sur un nombre d'étapes donné, mais jusqu'à que l'objet soit revenu à son point de départ. Effectuer plusieurs simulations (il n'est pas impossible que Python plante lors de ces essais) et commenter les résultats obtenus.