Le Mans Université, Master 1 actuariat

Python – TP3 : Numpy et entrées/sorties – 1h30

I Numpy

Consulter la documentation de Numpy (rubrique N-dimensional array) afin de répondre aux exercices suivants.

Exercice 1 :

Créer un ndarray de taille 10 dont tous les éléments sont nuls sauf la cinquième valeur qui vaut 1.

Exercice 2 :

Créer un ndarray contenant les entiers de 50 à 99.

Exercice 3 :

Inverser le ndarray vecteur des 100 premiers nombres entiers.

Exercice 4 :

Créer un ndarray de taille 10x10 contenant des valeurs aléatoires tirées uniformément et trouver les valeurs minimales et maximales.

Exercice 5 :

Créer l'array suivant par la programmation (sans rentrer les données à la main !)

1	6	11
2	7	12
3	8	13
4	9	14
5	10	15

Créer ensuite un autre array issu du premier contenant les lignes 3 et 4.

II Trois problèmes

Exercice 6 : Suite de Syracuse

On considère un nombre entier naturel non nul u_0. S'il est pair, on le divise par 2 ; s'il est impair, on le multiplie par 3 et on ajoute 1. On obtient une suite $(u_n)_{n\in\mathbb{N}}$ définie par

$$u_0 \in \mathbb{N}, \quad u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ pair} ; \\ 3u_n + 1 & \text{si } u_n \text{ impair.} \end{cases}$$

1. Écrire un algorithme qui demande en entrée la valeur u_0 et un nombre entier n, puis calcule et retourne un ndarray de dimension 1 composé des termes $u_0, u_1, \ldots, u_n$.

La conjecture de Syracuse affirme que pour tout u_0 il existe un indice n tel que $u_n = 1$.

2. Reprendre l'algorithme précédent pour créer une fonction `vol` ayant pour paramètre a qui retourne l'array L composé de tous les termes $u_0, u_1, \ldots, u_n$, où $u_0 = a$ et n est le premier entier tel que $u_n = 1$.

La liste `vol`(a) est le vol de la suite. Le temps de vol est le plus petit entier n tel que $u_n = 1$. L'altitude est le plus grand terme de L.

3. Écrire une fonction nommée `Tvol` ayant pour paramètre a qui retourne pour la suite avec $u_0 = a$, son vol et son temps de vol.
4. Représenter graphiquement le nuage de points (k, u_k) pour k allant de 0 jusqu'au temps de vol.
5. Déterminer le plus grand temps de vol de S_n pour $n \leq 100000$.

Exercice 7 : Loi uniforme

Les premiers algorithmes de simulation de la loi uniforme étaient basés sur la congruence (Lehmer, 1950). Le principe est le suivant :

— Utiliser trois paramètres entiers a, c et m et une valeur initiale y_0 (appelée `seed`) ;

— Créer une suite d'entiers

$$y_{n+1} = (ay_n + c) \bmod m$$

compris entre 0 et m ;

— Ramener les valeurs entre 0 et 1 : $x_n = y_n/m$.

1. Écrire une fonction Python ayant pour paramètres $N \in \mathbb{N}$, a, c, m et y_0, qui renvoie la liste des N premières réalisations de y_i et de x_i.
2. La tester avec $a = 13$, $c = 0$, $m = 31$ et $y_0 = 1$. Tracer le nuage de points (x_i, x_{i+1}), $i = 0, \ldots, N-1$. Que remarque-t-on ?

Dans les années 60, sur IBM, Scientific Subroutine Package (SSP) utilisait $a = 65539$, $c = 0$, et $m = 2^{32}$.

3. Tracer le nuage de points (x_i, x_{i+1}), $i = 0, \ldots, N-1$ et la fonction de répartition empirique des x_i. Que pensez-vous de la qualité de la simulation de la loi uniforme ?
4. Vérifier que $x_{k+2} - 6x_{k+1} + 9x_k$ sont tous entiers. Ainsi

$$-6 < x_{k+2} - 6x_{k+1} + 9x_k < 10.$$

Quel problème cela implique-t-il pour les simulations ?

5. Pour visualiser cet effet, recopier le code

```
from mpl_toolkits.mplot3d import Axes3D
fig=plt.figure(2)
ax = fig.add_subplot(111,projection='3d',elev = 10, azim =
    58)
ax.scatter(x[:N-2],x[1:N-1],x[2:])
```

À partir des années 1990, l'algorithme appelé "standard minimal" choisit : $a = 7^5 = 16807$, $c = 0$, $m = 2^{31} - 1 = 2147483647$.

6. Reprendre la question 3 avec ces chiffres.
7. Utiliser la fonction `kstest` du module `scipy.stats` pour tester si les simulations suivent bien la loi uniforme.

Exercice 8 : Matrice et image

Le module `numpy` permet de gérer les matrices via `numpy.array`.

1. Définir la matrice $T = \begin{pmatrix} -2 & 5 \\ 9 & 0 \end{pmatrix}$.
2. Calculer sa transposée, son déterminant et son inverse.
3. Le code suivant permet de récupérer une image

```
from scipy import misc
Image=misc.ascent()
```

   Quelle est la taille de cette image (utiliser `shape`) ?
4. Visualiser la avec la commande `imshow` de `matplotlib.pyplot`.
5. Calculer la transposée de Image et visualiser la.