Université Lyon 1 Année 2018-2019
Master Mathématiques Générales 1ère année
Logiciels Scientifiques

TP4 – Python

Pour tout le TP, on pourra entrer les commandes

```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt
```

1 Aléatoire

Exercice 1 - Fonctions utiles

Tester les commandes suivantes et afficher en le résultat.

```
M=np.array([[1, 3, 5], [-2, 6, 1]])
M>2
M[M>2]
indices=np.nonzero(M>2)

a=-2; b=2; h=0.5
x=np.arange(a, b+h/2, h)
y=np.linspace(a, b, round((b-a)/h+1))

l=np.logical_and(x>-1, x<1)
x[l]=3.14

np.cumsum(x)
np.sum(x)
np.mean(x)

poids=np.ones(x.size)
poids[0::2]=2
np.average(x)
np.average(x, weights=poids)

np.sort(x)
-np.sort(-x)
np.flip(np.sort(x),axis=0)
np.flipud(np.sort(x))

uni_x,num_x=np.unique(x, return_counts=True)
```

Exercice 2 - Génération d'aléa

Entrer les commandes suivantes et aller voir l'aide pour chacune d'entres elles.

```
npr.rand()
npr.rand(2, 4)
npr.randint(-5, 10, (2, 10))
a=-1; b=1;
npr.uniform(a, b, 1000)
npr.exponential(2,1000)
mu=1; sigma=2;
npr.normal(mu, sigma, 1000)
n=5; p=0.2;
npr.binomial(n, p, 1000)
```

La fonction `npr.rand` dépend d'une "graine", une suite (u_n) initialisée à l'ouverture de Python. À graine fixée, `npr.rand` fournira toujours la même série de valeurs.
La commande `npr.rand('seed')` affiche le terme u_n courant. La commande `rand('seed',k)` impose $u_n = u_k$, ce qui permet de répéter à l'identique un tirage aléatoire et contrôler ainsi ses résultats.

```
npr.seed(10);
npr.rand()
npr.rand()

npr.seed(10)
npr.rand()
```

Exercice 3 - Affichage

```
N=1000
tab_n=np.arange(1, N+1)
plt.plot(tab_n, np.cumsum(npr.rand(N))/tab_n)
plt.show()

n=10; p=0.2
u=npr.binomial(n, p, N)
x=np.sort(u)
y=tab_n/N
plt.plot(x, y)
plt.axis([-0.5, n, 0, 1])
plt.show()

U=npr.rand(2, N)
plt.plot(U[0, :], U[1, :], 'r.')
plt.show()
```

La commande `npr.hist` permet de tracer des histogrammes (voir l'aide pour les options).

```
N=1000
L=npr.normal(2, 1, N)

plt.hist(L, bins=100)
plt.show()
plt.hist(L, bins='auto')
plt.show()
plt.hist(L, bins='auto', density=True)
plt.show()
```

2 Exercices d'application

Exercice 1 - Lancé de dés

1. Écrire une fonction `dice()` simulant le résultat du lancer d'un dé non pipé à six faces.
2. Tester la validité de cette fonction en calculant les fréquences de sortie de chacun des chiffres au bout de $N = 100000$ tirages.

Exercice 2 - Paradoxe du chevalier de Méré

Le chevalier de Méré (1607-1684) qui était un grand joueur, avait remarqué que, lorsqu'on joue au dé, parier sur l'apparition d'un 6 en lançant 4 fois le dé était avantageux. Par un argument d'homothétie, il avait conclu que parier sur l'apparition d'un double-six quand on lance 24 fois deux dés l'était aussi.

1. En effectuant un grand nombre de simulations, vérifier que le premier jeu est avantageux.
2. En effectuant un grand nombre de simulations, constater que le second jeu ne l'est pas avantageux.

Exercice 3 - Problème des anniversaires

Soit une assemblée de p individus dont aucun n'est né un 29 février.

1. Écrire une fonction `anniv(p)` qui retourne 1 si deux individus dans une assemblée de p personnes sont nés le même jour, et 0 sinon.
2. Écrire une fonction `ProbAnniv(N,p)` qui retourne l'approximation de la probabilité d'avoir eu deux individus nés le même jour dans une assemblée de p personnes pour N simulations, et qui trace l'évolution au cours du calcul de l'approximation en fonction du nombre d'expériences déjà réalisées.
3. Pour $N = 1000$, effectuer les simulations avec $p = 24$, puis $p = 35$

Exercice 4 - Mouvement brownien

Le mouvement brownien, ou processus de Wiener, est une description mathématique du mouvement aléatoire d'une "grosse" particule immergée dans un fluide et qui n'est soumise à aucune autre interaction que des chocs avec les "petites" molécules du fluide environnant.
On simule ce mouvement en considèrant une particule se déplaçant suivant un axe (Ox). Aux instants $t_k = k\Delta t$, elle effectue un déplacement X_k. Chaque déplacement X_k est une variable aléatoire suivant une loi uniforme $U[-1, 1]$. On note $R_n = R(n\Delta t)$ la position de la particule après n intervalles Δt. On notera $T = N\Delta t$ le temps final de la simulation.

1. Écrire un programme `brownien(R0,n)` simulant la position R_n d'une particule avec $R_0 = 0$ et $n = 10000$.
2. Tracer une trajectoire correspondant à une réalisation du programme précédent.
3. On considère $p = 2000$ réalisations du programme précédent.
   (a) Tracer sur une même figure la moyenne arithmétique $\langle R_n \rangle$ et la moyenne quadratique $\sqrt{\langle R_n^2 \rangle}$ de la position des particules pour chaque pas de temps sur les p réalisations.
   (b) Vérifier que $\langle R_n^2 \rangle$ est proportionnel au temps.
   (c) Tracer sur une meme figure l'histogramme des R_N pour les p réalisations. Que constatez-vous ?
4. Modifier le programme `brownien` pour simuler la position R_n d'une particule dans $\mathbb{R}^2$ en prenant $R(0) = (0, 0)$ pour point de départ.
5. Tracer une trajectoire correspondant à une réalisation du programme $2D$ précédent.
