NOM : *1h*

DS 8 - CB

THÈME : INFORMATIQUE

On suppose importé dans la suite les bibliothèques `numpy`, `numpy.random`, `matplotlib.pyplot`, respectivement avec les alias `np`, `rd` et `plt`. On rappelle aussi que l'on code une matrice rectangulaire par `A=np.array([[ligne 1], ... [dernière ligne]])` et le coefficient en position (i, j) s'obtient par `A[i,j]`.

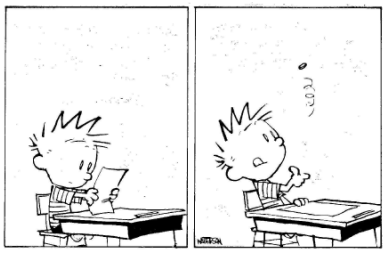
Exercice 1

On lance n fois une pièce équilibrée, et on note X_n le nombre de fois où l'on obtient deux résultats identiques consécutifs.

1. Écrire un programme `SimuX` qui prend en argument n et simule la variable aléatoire X_n.

2. Utiliser ensuite le programme `SimuX` pour estimer la valeur de $\mathbf{E}(X_n)$.

3. Dans une infinité de tirages d'une pièce équilibrée. Comment simuler la variable aléatoire qui renvoie le rang d'apparition de la première succession de Pile-Pile (exemple si on tire PFFPPFP, on renvoie 4) ?



Calvin et Hobbes

Problème A : Suites de Fibonacci aléatoires

- *Le cas déterministe*
  On considère la suite $(f_n)_{n\in\mathbb{N}}$ définie par : $f_0 = 0, \quad f_1 = 1 \quad$ et $\quad \forall n \in \mathbb{N}, \quad f_{n+2} = f_{n+1} + f_n$.

4. Modifier le programme suivant qui prend en argument n et renvoie la matrice ligne

$$L_n = \begin{bmatrix} f_0 & f_1 & f_2 & \cdots & f_n \end{bmatrix}.$$

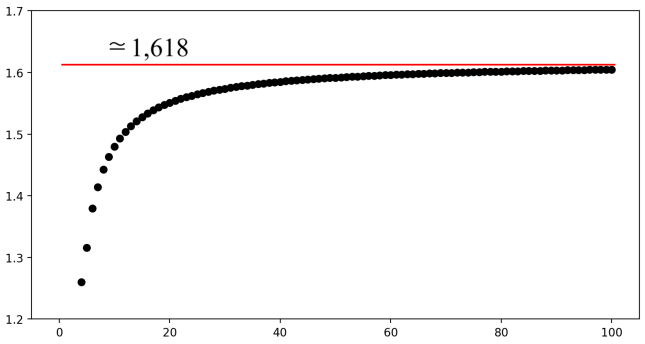
Editeur

```
def Fibo(n):
    L= ...
    L[1]=  ...
    for i in range( ...  ):
        ...
    return L
```

5. Que permet de conjecturer le code et résultat suivants ?

Editeur

```
n=100; L=Fibo(n); N=np.zeros(n)
for i in range(1,n):
    N[i]=L[i]**(1/i)
NbreOr=(1+np.sqrt(5))/2
plt.ylim(1.2,1.7)
 # pour restreindre l'affichage en
    ordonnée
plt.plot([0,n],[NbreOr,NbreOr],'r')
plt.plot(np.linspace(1,n,n),N,'ko')
plt.show()
```



- *Le cas aléatoire*
  Soient $p \in [0;1]$ et $(U_n)_{n\in\mathbb{N}}$, $(V_n)_{n\in\mathbb{N}}$, deux suites de variables aléatoires discrètes définies sur le même espace probabilisé, indépendantes et de même loi que la variable U

$$\mathbf{P}(U = -1) = 1 - p \quad \text{et} \quad \mathbf{P}(U = 1) = p.$$

On définit alors la suite de variables aléatoires $(F_n)_{n\in\mathbb{N}}$ par la récurrence

$$F_0 = 0, \quad F_1 = 1 \quad \text{et} \quad \forall n \in \mathbb{N}, \quad F_{n+2} = U_n \cdot F_{n+1} + V_n \cdot F_n.$$

On pose de plus les matrices aléatoires :

$$\forall n \in \mathbb{N}, \quad A_n = \begin{bmatrix} 0 & V_n \\ 1 & U_n \end{bmatrix} \quad \text{et} \quad L_n = \begin{bmatrix} F_n & F_{n+1} \end{bmatrix}.$$

6. Donner une relation simple entre A_n, L_n et L_{n+1}.

7. En déduire l'existence d'une matrice B_n, que l'on exprimera sous forme de produit telle que $L_n = L_0 B_n$.

8. Écrire un programme `SimuU` qui prend en argument p puis simule la variable U ? En déduire un second programme qui simule la matrice A_n.

9. Comment simuler la matrice B_n ? La variable F_n ? Comment tester ce dernier programme avec $p = 1$?
On rappelle que le produit matriciel en python peut s'obtenir par la commande `np.dot( , )`.

10. Écrire un programme qui prend en argument une matrice carrée A de taille n et renvoie sa norme définie comme la racine carrée de la somme des coefficients de la matrice au carré (si $A = (a_{i,j})$, $\|A\| = \sqrt{\sum_{i,j=1}^{n} {a_{i,j}}^2}$).

11. Un théorème des mathématiciens Furstenberg et Kesten affirme que la norme matricielle de B_n croît approximativement en λ^n pour un certain λ. Proposer une méthode pour vérifier numériquement cette affirmation.

Problème II : Simulations d'une loi de Pareto

Soient $k \in \mathbb{N}^*$ et $\lambda \in \mathbb{R}_*^+$. On montre que la fonction suivante est une densité de probabilité : $f_{\lambda,k} : x \in \mathbb{R} \mapsto \begin{cases} 0 & \text{si } x \leqslant \lambda \\ \frac{k\lambda^k}{x^{k+1}} & \text{sinon.} \end{cases}$

On dit que X suit la loi de Pareto de paramètres λ et k si X admet $f_{\lambda,k}$ comme densité.

12. Déterminer la fonction de répartition $F_{\lambda,k}$ d'une variable suivant la loi de Pareto de paramètres λ et k.

13. *Méthode 1 (par inversion).*
On rappelle que si U *suit la loi uniforme sur* $]0;1[$ *et* X *est une variable aléatoire à densité dont la fonction de répartition F est une bijection de* $]a;b[$ *avec* $-\infty \leqslant a < b \leqslant +\infty$ *sur* $]0;1[$*, alors la variable* $Y = F^{-1}(U)$ *suit la même loi que* X.
À l'aide du rappel, écrire une fonction `Pareto` qui prend en arguments λ et k puis simule une variable aléatoire de loi de Pareto de paramètres λ et k.

14. *Méthode 2.*
Soient $X_1, \dots, X_k$, k variables aléatoires mutuellement indépendantes et toute de même loi uniforme sur $]0;1]$. On pose alors

$$Y = \frac{\lambda}{\max(X_1, \dots, X_k)}.$$

Montrer que Y suit une loi de Pareto de paramètres λ et k.

15. En déduire une autre méthode pour simuler une loi de Pareto.

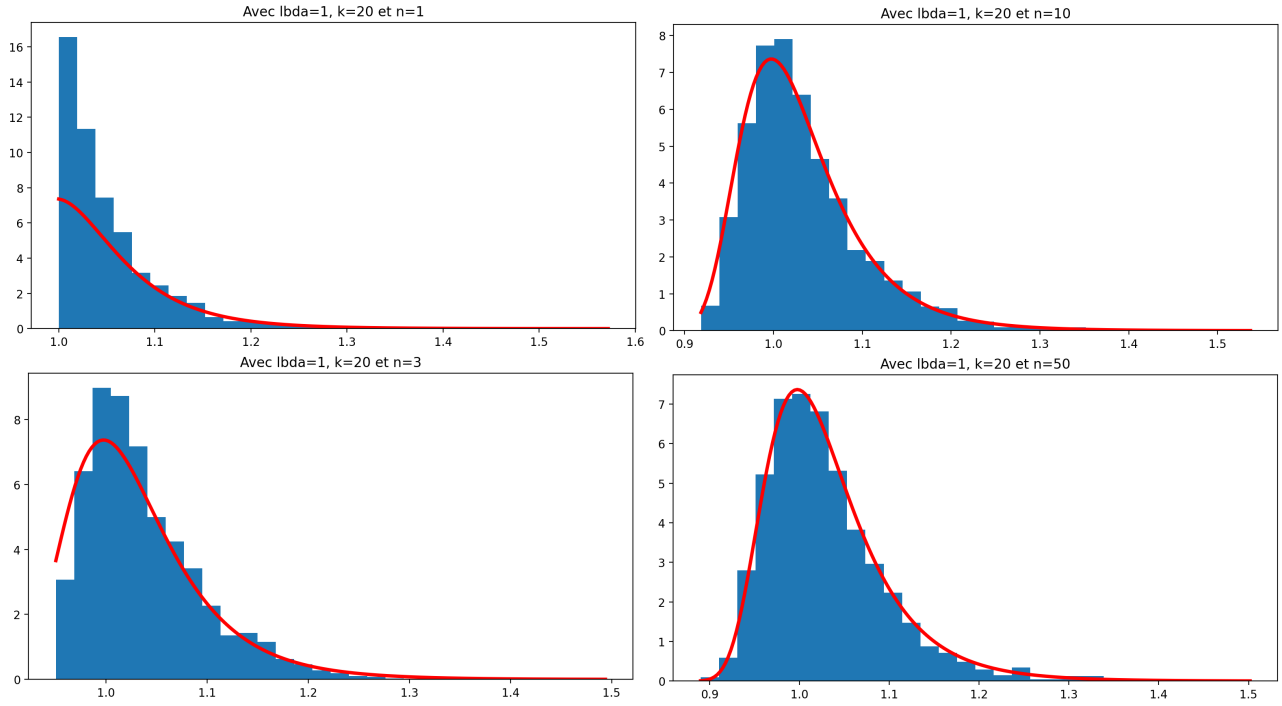
16. Pour $(X_i)_{i\in\mathbb{N}^*}$ une suite de variable aléatoire de loi de Pareto $\lambda = 1$, $k = 20$, on pose

$$M_n = n^{-1/k} \max(X_1, X_2, \dots, X_n).$$

Le code suivant permet de conjecturer une convergence en loi. Laquelle ? (en bonus la preuve si vous avez le temps).

Editeur

```
def Mn(n,k):
    m=2000
    Ech=np.zeros(m)
    for i in range(m):
        A=np.zeros(n)
        for j in range(n):
            A[j]=Pareto(1,k)
        Ech[i]=n**(-1/k)*np.max(A)
    plt.clf()
    plt.hist(Ech,30,density='True')
    x=np.linspace(min(Ech),max(Ech),200)
    y=k*x**(-k-1)*np.exp(-1/x**k)
    plt.plot(x,y,'r',linewidth=3)
    Titre='Avec lbda=1, k='+str(k)+' et n='+str(n)
    plt.title(Titre)
    plt.show()
```



– FIN –

DS 8 - solution

Exercice

1.

```
def simuX(n):
    PF=rd.rand(n)<(1/2)
    C=0
    for i in range(n-1):
        if PF[i]==PF[i+1]:
            C=C+1
    return C
```

2.

```
def approx(n):
    e=0
    for i in range(5000):
        e+=simuX(n)
    return e/5000
```

On peut tester pour $n=2$, $n=3$ car on a facilement accès aux lois de X_2 et X_3.

```
    >>> approx(3)
0.999

>>> approx(2)
0.5022
```

3.

```
def simuY():
    n=1
    LancerOld=rd.rand()<1/2
    LancerNew=rd.rand()<1/2

    while LancerOld!=LancerNew:
        n+=1
        LancerOld=LancerNew
        LancerNew=rd.rand()<1/2

    return n
```

Problème A

4.

```
def Fibo(n):
    L=np.zeros(n+1)
    L[1]=1
    for i in range(n-1):
        L[i+2]=L[i+1]+L[i]
    return L
```

5. Le code affiche les 100 premiers termes $(f_n^{1/n})_{n\in[\![1;100]\!]}$. On conjecture que

$$f_n^{1/n} \underset{n\to\infty}{\longrightarrow} \varphi = \frac{1+\sqrt{5}}{2}.$$

Ce résultat est assez facile à démontrer car, à partir des résultats sur les suites récurrentes linéaires d'ordre 2, on a

$$F_n = \frac{1}{\sqrt{5}}\left(\varphi^n - \varphi'^n\right),$$

$$\text{avec} \quad \varphi = \frac{1+\sqrt{5}}{2} \quad \text{et} \quad \varphi' = \frac{1-\sqrt{5}}{2} = -\frac{1}{\varphi}.$$

Le nombre φ est appelé le nombre d'or. Ce nombre a de nombreuses propriétés arithmétiques. Par exemple :

$$\varphi = \cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{\ddots}}}}.$$

L'Antiquité attribue à ce nombre des propriétés « esthétiques ». Un rectangle dont le rapport longueur sur largeur serait égal au nombre d'or serait le plus harmonieux. Citons plus récemment, l'utilisation faite par l'architecte Le Corbusier avec le modulor permettant de construire des habitations à taille humaine.

6. Vérifier que pour tout $n\in\mathbb{N}$

$$L_nA_n = L_{n+1}.$$

Attention à l'ordre du produit.

7. Par récurrence, montrer que

$$B_n = A_0A_1\dots A_n = \prod_{i=0}^{n} A_i$$

convient.

8.

```
def simuU(p):
    return (-1)**(rd.rand()>p)

def simuAn(p):
    A=np.array([[0,simuU(p)],[1,simuU(p)
        ]])
    return A
```

9.

```
def simuBn(n,p):
    B=simuAn(p)
    for i in range(1,n):
        A=simuAn(p)
        B=np.dot(B,A)
    return B

def simuFn(n,p):
    B=simuBn(n,p)
    return B[1,0]
```

Pour $p = 1$, on doit retrouver le cas déterminé. Testons (on croise les doigts, suspense, est-ce que cela marche??!!).

```
>>> Fibo(18)[-1]
2584.0

>>> simuFn(18,1)
2584

Oui, ouf !
```

10.

```
def norme(A):
    [n,p]=np.shape(A)
    s=0
    for i in range(n):
        for j in range(p):
            s+=A[i,j]**2
    return s**(1/2)
```

11. On peut tester dans un premier temps si la suite de terme général

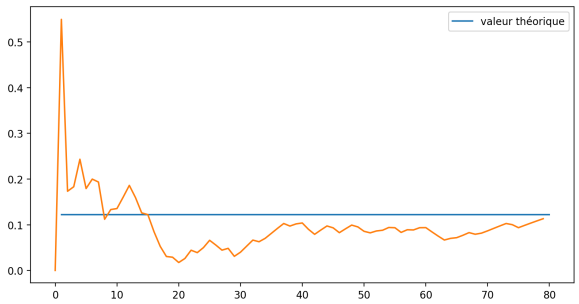
$$\frac{\ln(\|B_n\|)}{n}$$

converge vers une constante.

```
def cv(n,p):
    N=np.zeros(n)
    B=simuAn(p)

    for i in range(1,n):

        A=simuAn(p)
        B=np.dot(B,A)
        N[i]=np.log(norme(B))/i

    return N
```

Oui, on trouve :

```
l=np.log(1.131988)
plt.plot([1,80],[l,l], label='valeur théorique')
plt.legend()
plt.plot(cv(80,1/2))
plt.show()
```



Problème B

12. Par intégration

$$F_{\lambda,k} : x \mapsto \begin{cases} 0 & \text{si } x \leqslant \lambda \\ 1 - \frac{\lambda^k}{x^k} & \text{sinon} \end{cases}$$

13. On vérifie que $F_{\lambda,k}$ réalise une bijection de $]\lambda;+\infty[$ dans $]0;1[$ avec pour application réciproque

$$x \in]0;1[\mapsto \lambda(1-x)^{-1/k}.$$

On en déduit le code

```
def Pareto(lbda,k):
    u=rd.random()
    return lbda*(1-x)**(-1/k)
```

14. Passer par la fonction de répartition...

15.

```
def Pareto2(lbda,k):
    Uni=rd.rand(k)
    return lbda/np.max(Uni)
```

16. On conjecture une convergence en loi vers une variable aléatoire à densité dont une densité est nulle sur $\mathbb{R}^-$ et pour $x \in \mathbb{R}_*^+$

$$f(x) = kx^{-k-1}\exp(-1/x^k).$$