TD26

Révisions de Python

Programmation

Exercice 1 (★ - Sur la série harmonique)

1. Écrire un programme qui prend comme paramètre un entier n et retourne $u_n = \sum_{k=1}^{n} \frac{1}{k} - \ln(n)$.

   Vérifier alors numériquement que la suite (u_n) est convergente, et donner une valeur approchée de sa limite.

2. Écrire une fonction qui prend comme paramètre un réel $A > 0$ et qui retourne la valeur du plus petit entier n tel que $\sum_{k=1}^{n} \frac{1}{k} > A$.

   Quel est le plus petit entier n tel que $\sum_{k=1}^{n} \frac{1}{k} > 15$?

Exercice 2 (★★ - Série harmonique alternée)

On considère la série harmonique alternée $\sum_{n\geq 1} \frac{(-1)^{n-1}}{n}$. On note $S = (S_n)_{n\in\mathbb{N}^*}$ la suite de ses sommes partielles.

1. Écrire une commande définissant un vecteur ligne `u` tel que pour tout $1 \leq i \leq 50$, `u[i-1]` soit égal à $\frac{(-1)^{i-1}}{i}$.

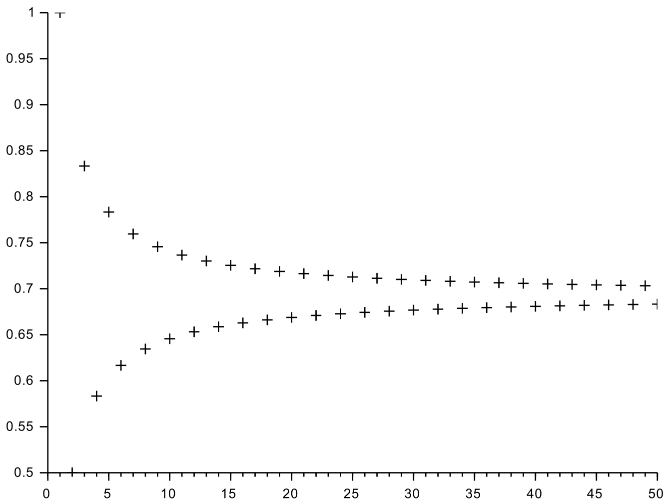
2. Écrire une commande définissant un vecteur ligne `v` tel que pour tout $1 \leq i \leq 50$, `v[i-1]` soit égal à $\sum_{k=1}^{i} \frac{(-1)^{k-1}}{k}$.

3. En exécutant la commande :

   `plt.plot(np.arange(1,51)`

   on obtient le graphe ci-contre.

   Que peut-on dire des suites extraites (S_{2n}) et (S_{2n+1}) ? de la série S ?



4. On admet que $\sum_{n=1}^{+\infty} \frac{(-1)^{n-1}}{n} = \ln(2)$. Écrire un script demandant une valeur $\varepsilon > 0$ à l'utilisateur, et qui renvoie le plus petit entier naturel n pour lequel $|S_n - \ln(2)| \leq \varepsilon$.

Exercice 3 (★★)

Soit (u_n) la suite définie par récurrence par $u_0 = \frac{1}{2}$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = u_n^2 - u_n \ln(u_n)$.

1. Écrire une fonction `suite` qui prend comme paramètre un entier n et renvoie la valeur de u_n correspondante.
2. On se propose de vérifier graphiquement que la suite (u_n) est croissante et tend vers 1.

   Écrire à cet effet un programme qui trace un graphique sur lequel se trouvent les points (n, u_n), pour $n \in [\![0, 100]\!]$.
3. Écrire une fonction nommée `plus_petit_n` qui prend comme paramètre un entier p et retourne la plus petite valeur de n pour laquelle $|1 - u_n| < 10^{-p}$.

Exercice 4 (★★)

Soit (u_n) la suite définie par $u_0 = u_1 = u_2 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+3} = u_{n+2} - 2u_{n+1} + u_n$.
Écrire une fonction d'entête `def suite_rec(n)` qui prend comme paramètre $n \in \mathbb{N}$ et retourne la valeur de u_n.

Exercice 5 (★★)

Soient (a_n) et (b_n) les suites définies par $a_0 = 2$, $b_0 = 3$ et :

$$\forall n \in \mathbb{N}, \quad a_{n+1} = \sqrt{a_n b_n} \quad \text{et} \quad b_{n+1} = \frac{a_n + b_n}{2}.$$

Écrire une fonction d'entête `def suites(n)` prenant comme paramètre un entier n et retournant les valeurs de a_n et b_n correspondantes.
Constater que ces suites convergent vers une même limite.

Exercice 6 (★★★ - Extrait d'Edhec 2019)

Dans cet exercice, n désigne un entier naturel non nul.

1. On admet que, si a et b sont des entiers tels que $a < b$, la commande `rd.randint(a,b)` permet à `Python` de simuler une variable aléatoire suivant la loi uniforme discrète sur $[\![a, b-1]\!]$.
   Compléter le script suivant pour que les lignes (5), (6), (7) et (8) permettent d'échanger les contenus des variables $A(j)$ et $A(p)$.

```
1  n=int(input('entrez une valeur pour n :'))
2  A=np.arange(1,n+1)
3  p=n-1
4  for k=1:n
5          j=rd.randint(0,p+1)
6          aux=----
7          A[j]=----
8          A[p]=----
9          p=p-1
10 print(A)
```

2. On suppose dorénavant qu'après exécution du script précédent correctement complété, le vecteur A est rempli de façon aléatoire par les entiers de $[\![1, n]\!]$ de telle sorte que les $n!$ permutations soient équiprobables.
   On considère alors les commandes `Python` suivantes (exécutées à la suite du script précédent) :

```python
m=A[0]
c=1
for k in range(1,n):
    if A[k]>m :
        m=A[k]
        c=k
print(c)
```

(a) Expliquer pourquoi, à la fin de la boucle `for`, la variable `m` contient la valeur n.

(b) Quel est le contenu de la variable c affiché à la fin de ces commandes ?

On admet que les contenus des variables `A[0]`, `A[1]`, ..., `A[n-1]` sont des variables aléatoires notées $A_1,\ A_2,\ \dots,\ A_n$ et que le nombre d'affectations concernant la variable informatique c effectuées au cours du script présenté au début de la question 2., y compris la première, est aussi une variable aléatoire, notée X_n.
On suppose que ces variables aléatoires sont toutes définies sur le même espace probabilisé $(\Omega, \mathscr{A}, P)$.

3. Donner la loi de X_1.

4. (a) Montrer que $X_n(\Omega) = [\![1, n]\!]$.
   (b) Déterminer $P(X_n = 1)$ et $P(X_n = n)$. En déduire les lois de X_2 et X_3.
   (c) En considérant le système complet d'événements $((A_n = n), (A_n < n))$, montrer que :
   $$\forall n \geq 2,\ \forall j \in [\![2, n]\!],\ P(X_n = j) = \frac{1}{n}P(X_{n-1} = j-1) + \frac{n-1}{n}P(X_{n-1} = j)$$
   (d) Donner la loi de X_4.

Simulation de variables aléatoires

Exercice 7 (★)

En utilisant `rd.random()`, mais sans `rd.binomial`, écrire une fonction d'entête `def binomiale(n,p)` qui simule une réalisation d'une loi $\mathscr{B}(n, p)$.
Indication : on se rappellera qu'une loi binomiale correspond au nombre de succès lors d'une répétition d'épreuves de Bernoulli indépendantes.

Exercice 8 (★★)

1. Compléter la fonction suivante pour qu'elle simule une réalisation d'une loi géométrique de paramètre p.

```python
def geom(p):
    y = 1
    while rd.random() ----- :
        y = -----
    return( ----- )
```

2. On dit qu'une variable suit la loi binomiale négative de paramètres n et p si elle a la même loi que $\displaystyle\sum_{i=1}^{n} X_i$ où les X_i sont des variables i.i.d. suivant la loi $\mathscr{G}(p)$.

   Écrire un programme d'entête `def bin_neg(n,p)` qui simule une réalisation d'une loi binomiale négative de paramètres n et p.

Exercice 9 (★★)

Une urne contient initialement des boules numérotées de 2 à n. On effectue un tirage dans cette urne, et on enlève de l'urne toutes les boules portant un numéro supérieur ou égal à celui de la boule tirée. On ajoute alors la boule numéro 1 dans l'urne, et on effectue un nouveau tirage, et on note X le numéro de la boule obtenue.
Écrire un programme qui simule la variable aléatoire X.

Exercice 10 (★★ - Ecricome 2016)

Soient a, b deux entiers strictement positifs. Une urne contient initialement a boules rouges et b boules blanches. On effectue une succession d'épreuves, chaque épreuve étant constituée des trois étapes suivantes :

- on pioche une boule au hasard dans l'urne,
- on replace la boule tirée dans l'urne,
- on rajoute dans l'urne une boule de la même couleur que celle qui vient d'être piochée.

Après n épreuves, l'urne contient donc $a + b + n$ boules. Pour tout $n \in \mathbb{N}^*$, on note X_n le nombre de boules rouges qui ont été **ajoutées** dans l'urne (par rapport à la composition initiale) à l'issue des n premières épreuves.

On souhaite simuler l'expérience grâce à `Python`.

1. Compléter la fonction suivante, qui simule le tirage d'une boule dans une urne contenant x boules rouges et y boules blanches et qui retourne la valeur 0 si la boule est rouge et 1 si elle est blanche.

```python
def tirage(x,y):
    r = rd.random()
    if ............ :
        res = 0
    else :
        res = 1
    return res
```

2. Compléter la fonction suivante, qui simule n tirages successifs dans une urne contenant initialement a boules rouges et b boules blanches (selon le protocole décrit ci-dessus) et qui retourne la valeur de X_n :

```python
def experience(a,b,n):
    x = a
    y = b
    for k in range(n):
        r = tirage(x,y)
        if r == 0 :
            x = ........
        else :
            ............
    Xn = ........
    return Xn
```

3. Écrire une fonction `simulation(a,b,n,m)` qui fait appel `m` fois à la fonction précédente pour estimer la loi de X_n. Le paramètre de sortie sera un vecteur à $n+1$ composantes contenant les approximations de $P(X_n = 0)$, $P(X_n = 1), \dots, P(X_n = n)$.

4. On s'intéresse au cas où $a = b = 1$. On rappelle les commandes suivantes :

   - Si `x` et `y` sont des vecteurs de même taille, `plt.bar(x,y)` trace le diagramme en bâtons d'abscisses contenues dans `x` et d'ordonnées dans `y`.
   - `plt.figure(n)` ouvre une fenêtre graphique et trace la figure `n` dans celle-ci.

   On exécute le code suivant :

```python
for n in range(1,6):
    x = np.arange(n+1)
    y = simulation(1,1,n,100000)
    plt.figure(n)
    plt.bar(x,y)
plt.show()
```

   On obtient les figures suivantes :

Figure 1

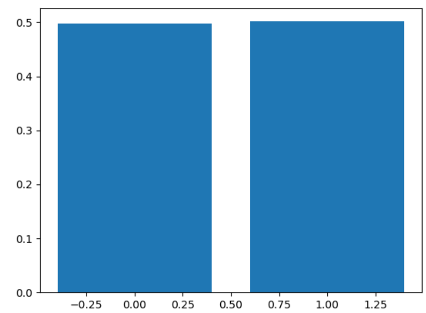


Figure 2

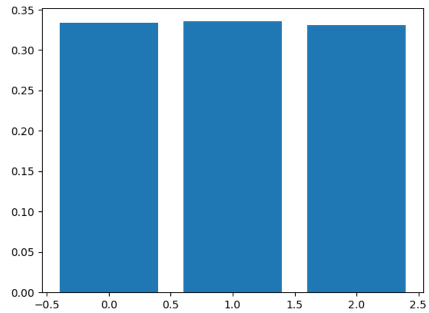


Figure 3

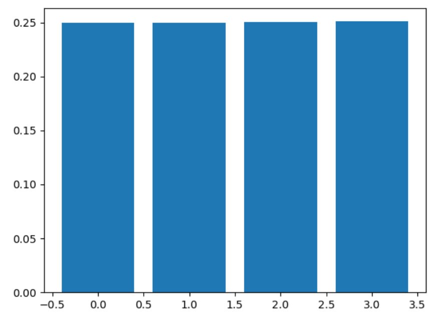


Figure 4

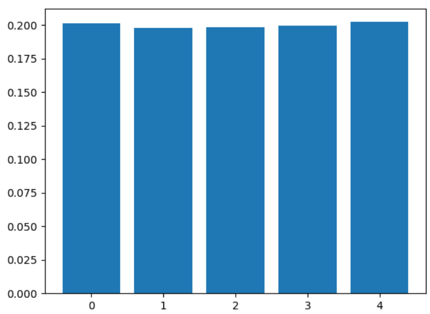
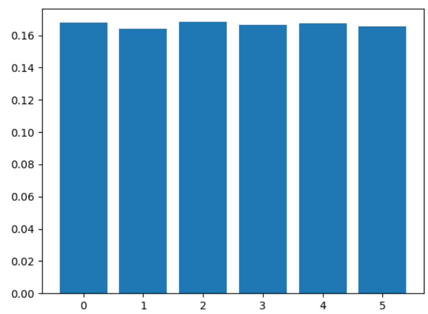


Figure 5



   À l'aide de ces résultats, conjecturer la loi de X_n.

Exercice 11 (★★)

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

Écrire un programme qui simule la variable aléatoire X. Utiliser ensuite ce programme pour estimer la valeur de $E(X)$.

Exercice 12 (★★)

Si $(X_1, \dots, X_n)$ sont des variables aléatoires indépendantes suivant la loi $\mathscr{N}(0,1)$, la loi suivie par la variable aléatoire $\displaystyle\sum_{i=1}^{n} X_i^2$ est appelée loi du χ^2 (prononcer « chi-deux ») de paramètre n.

1. Écrire une fonction `def chi2(n)` qui prend en paramètre un entier $n \geq 1$, et qui simule une variable suivant la loi $\chi^2(n)$.
2. Écrire une fonction qui simule la variable T_p , où $T_p = \max(Y_1, \dots, Y_p)$, où $Y_1, \dots, Y_p$ sont p variables aléatoires indépendantes suivant la loi $\chi^2(n)$.
3. Proposer une méthode pour obtenir une valeur approchée de $E(T_p)$.

Exercice 13 (★★ - Loi de Rayleigh)

1. Soit $\sigma > 0$. Montrer que la fonction $f : x \mapsto \begin{cases} \dfrac{1}{\sigma} x e^{-\frac{x^2}{2\sigma}} & \text{si } x \geq 0 \\ 0 & \text{sinon.} \end{cases}$ est une densité de probabilité.

   La loi d'une variable aléatoire admettant une telle densité est appelée loi de Rayleigh de paramètre σ. Dans toute la suite, on note X une variable aléatoire suivant une telle loi.

2. Donner la fonction de répartition F_X de X.
3. (a) Montrer que F_X réalise une bijection de $\mathbb{R}_+^*$ dans $]0,1[$. Déterminer une expression explicite de F_X^{-1}.

   (b) Soit U une variable aléatoire suivant une loi $\mathscr{U}(]0,1[)$. Montrer que $F_X^{-1}(U)$ suit la même loi que X.

   (c) Écrire une fonction `Python` d'entête `def rayleigh(sigma,n)` qui, étant donné un réel $\sigma > 0$ et un entier $n \in \mathbb{N}^*$, simule n réalisation d'une loi de Rayleigh de paramètre σ.

   (d) Vérifier la pertinence de cette simulation en comparant l'histogramme des fréquences et la densité, puis les fonctions de répartitions empirique et théorique.
4. Estimer numériquement l'espérance et l'écart-type d'une variable aléatoire suivant une loi de Rayleigh de paramètre σ. Retrouver ces résultats par le calcul.
5. (a) Si X suit une loi de Rayleigh de paramètre σ, quelle est la loi de $Y = X^2$.

   (b) À l'aide de la question précédente, proposer une nouvelle fonction `Python` d'entête `def rayleigh2(sigma)` qui, étant donné un réel $\sigma > 0$, simule une réalisation d'une loi de Rayleigh de paramètre σ.

Exercice 14 (★★★ - QSP HEC 2016)

Donner la finalité du programme suivant :

```python
N = 100000 ; S = 0
for i in range(N):
      u = rd.random()
      S = S + (4/N)*1/(1+u**2)
print(S)
```

On pourra penser à la loi faible des grands nombres.

Exercice 15 (★★★ - QSP HEC 2021)

On considère le code `Python` suivant :

```python
def g(x):
    X = (1/2)*rd.random(10000)
    Y = (1/3)*rd.random(10000)
    S = 0
    for k in range(10000):
        if X[k]+Y[k] <= x :
            S = S+1
    return S/10000

def graph2():
    x = np.linspace(-0.1,1,101)
    y = g(x)
    plt.plot(x,y)
    plt.show()
```

1. Commenter les fonctions `Python`.
2. Dessiner le graphe de sortie de la fonction `graphe2`.

Exercice 16 (★★★★ - QSP HEC 2021)

On considère le code `Python` suivant :

```python
def smul1(n,p):
    X = rd.geometric(p,n)
    Z = np.max(X)
    T = np.zeros(Z)
    for k in range(n)
        if T[X[k]-1]==0 :
            T[X[k]-1] = 1
    return np.sum(T)

def smul2(n,p):
    X = rd.geometric(p,10000,n)
    Y = np.zeros((10000,1))
    for j in range(10000):
        Z = np.max(X[j,:])
        T = np.zeros(Z)
        for k in range(n):
            if T[X[j,k]-1]==0 :
                T[X[j,k]-1] = 1
        Y[j] = np.sum(T)
    m = np.mean(Y)
```

1. Commenter la fonction `smul1`. Quelles sont les valeurs possibles de sortie ? Donner la loi de la variable aléatoire `smul1(2,0.75)`.
2. Le résultat de `smul2(2,0.75)` est `1.3971`. Commenter ce résultat.

Exercice 17 (★★★★ - QSP HEC 2015)

Dans une classe de 30 élèves, on considère une expérience consistant d'abord à demander à chaque élève sa date d'anniversaire. La suite de l'expérience (simulée 1000 fois sur ordinateur) est décrite par le programme `Python` suivant :

```python
Nexp = 1000 ; Neleve = 30 ; test = 0 ;
for n in range(Nexp):
    anniv = np.zeros(Neleve)
    for i in range(Neleve):
        anniv[i] = np.floor(365*rd.random())+1
    anniv = np.sort(anniv) # np.sort = tri par ordre croissant
    ok = 0
    for j in range(Neleve-1):
        if anniv[j] == anniv[j+1] :
            ok = 1
    test = test + ok
print(test/Nexp)
```

1. Le code retourne une valeur (à chaque fois différente) autour de $0,71$. Que représente cette valeur ?
2. Calculer la valeur exacte de la probabilité simulée par ce programme.
3. Écrire un programme `Python` permettant de déterminer le nombre d'élèves à partir duquel cette valeur dépasse 0.5.
