Révisions 10

Sujet d'Informatique

Exercice 1 (★)

On effectue une succession de lancers d'un dé équilibré. On note X le nombre de lancers nécessaire pour avoir quatre "six" de suite.

1. Écrire de deux manière une instruction qui simule un lancer de dé (on mettra le résultat dans la variable `de`) : l'une en utilisant les commandes `rd.random` et `np.floor` et l'autre en utilisant la commande `rd.randint`.

2. Compléter la suite d'instructions suivante qui simule des lancers successifs d'un dé jusqu'à obtenir quatre "six" consécutifs :

```
nbr_six_succ = 0
while ..... :
    de = .....
    if de == 6 :
        nbr_six_succ = .....
    else :
        nbr_six_succ = .....
```

3. Modifier le programme précédent pour écrire une fonction d'entête `def nb_de_lancers()` qui simule l'expérience et renvoie la valeur prise par la variable aléatoire X.

4. Proposer un programme permettant d'estimer l'espérance de X par la méthode de Monte-Carlo sur un grand échantillon de la variable aléatoire X (un 1000-échantillon par exemple).

Exercice 2 (★★)

On possède 75 dés équilibrés et 25 dés pipés. Sur chaque dé pipé, il y a 3 faces "6", 2 faces "1" et une face portant le numéro "2".
On choisit un dé au hasard dans le lot, on le lance et on note X la variable aléatoire égale au résultat obtenu.

1. Écrire une fonction `deeq()` qui renvoie une simulation de lancers d'un dé équilibré.

2. Écrire une fonction `depipe()` qui renvoie une simulation de lancers d'un dé pipé.

3. Écrire une fonction `simulX()` qui simule l'expérience dans son ensemble et retourne la valeur de X.

4. Écrire un programme qui réalise 10000 simulations de X et trace l'histogramme des fréquences.

Exercice 3 (★)

Un individu se déplace sur une droite graduée et orientée. A l'instant 0, il part du point d'abscisse 0. Toutes les minutes, il se déplace d'une unité à droite ou à gauche avec la même probabilité.
On note X_n la position de l'individu après le n-ième pas.
On veut estimer la probabilité que l'individu soit éloigné de strictement plus de 10 unités du point de départ au bout de 100 pas.

1. A quelle catégorie déjà étudiée appartient la suite de variables aléatoires (X_n) ?

   Quelle est la loi de X_{n+1} sachant $(X_n = i)$?

2. Compléter et exécuter la fonction suivante qui prend en entrée le nombre n de déplacements de l'individu et renvoie en sortie la valeur x de la position X_n de l'individu à l'instant n :

```
def deplacements(n) :
    x = 0
    for k in range(n) :
        if ..... :
            x = .....
        else :
            x = .....
    return(x)
```

Afficher quelques simulations de la variable X_{100}.

3. Compléter le programme suivant qui, à l'aide de 10000 simulations de la variable X_{100}, calcule une estimation t de la probabilité que l'individu soit éloigné de strictement plus de 10 unités de son point de départ au bout de 100 pas :

```
e = 0
for k in range(10000) :
    x = deplacements(100)
    if np.abs(x)>10 :
        e = .....
t = .....
print(t)
```

Exercice 4 (★★)

On constate que, dans une certaine région, s'il pleut un jour donné, alors il y a une chance sur deux pour qu'il pleuve encore le lendemain ; et s'il ne pleut pas, il ne pleut pas non plus le lendemain trois fois sur quatre. Par un certain jour de pluie, numéroté 0, on décide de noter le temps qu'il fait. Pour tout entier naturel n, on note X_n la variable aléatoire égale à 1 s'il pleut le n-ième jour qui suit celui où l'on a débuté les observations, et 2 s'il ne pleut pas (X_0 est certaine égale à 1).

1. Donner le graphe probabiliste et la matrice de transition associés à cette chaine de Markov.

2. Écrire une fonction `Python` permettant de simuler la chaine (X_n) sous la forme d'un vecteur à n composante, n étant entré par l'utilisateur, la k-ième composante valant 1 s'il pleut le jour numéro k et 2 sinon.

3. Écrire des instructions permettant d'afficher la probabilité qu'il pleuve une semaine après la première observation (c'est-à-dire le jour 7).

4. (a) Écrire des instructions permettant de déterminer l'état stable de la chaine de Markov.
   (b) Proposer des instructions permettant de vérifier si cet état stable correspond à la loi limite de la suite (X_n).

Exercice 5 (★★)

1. Écrire un programme `bin(n, p)` qui, étant donné un entier naturel n non nul et un réel $p \in]0,1[$, retourne une simulation d'une variable aléatoire X qui suit la loi $\mathcal{B}(n,p)$. Cette fonction utilisera la commande `rd.random` et n'utilisera pas `rd.binomial`.

2. Écrire un script qui demande n et p à l'utilisateur, qui fait 10000 simulations de X et les stocke dans une variable `X` et qui trace l'histogramme associé à `X`.

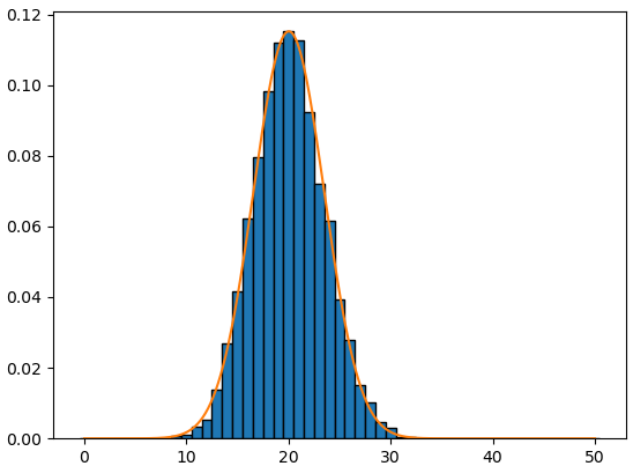
3. On ajoute à la suite les instructions suivantes :

```
def f(t, m, sigma):
    return(np.exp(-(t-m)**2/(2*sigma**2))/np.sqrt(2*np.pi*
sigma**2))

m = np.mean(X)
sigma = np.std(X)

T = np.linspace(0, n, 1000)
U = [f(t, m, sigma) for t in T]
plt.plot(T, U)
plt.show()
```

On obtient alors le graphique suivant (en prenant $n = 50$ et $p = 0.4$:



(a) Que représente f ?

(b) Quel théorème illustre le graphique obtenu ?

Exercice 6 (★★)

On considère une variable aléatoire X de loi uniforme sur $[0, a]$, où a est un réel strictement positif inconnu que l'on veut estimer. On dispose pour cela d'un échantillon $(X_1, \dots, X_n)$ de X et on veut comparer les deux estimateurs de a suivants :

$$T_n = \max(X_1, \dots, X_n) - \min(X_1, \dots, X_n) \quad \text{et} \quad Z_n = \frac{2}{n}\sum_{i=1}^{n} X_i.$$

1. (a) Écrire des instructions permettant de construire deux vecteurs `T` et `Z` contenant chacun respectivement 10 réalisations de T_{100} et Z_{100} pour une valeur de a entrée par l'utilisateur.

   (b) On obtient après exécution pour $a = 10$:

   - Pour T_{100} :
     `[9.744  9.258  9.661  9.951  9.929  9.806  9.536  9.565  9.786  9.729]`
   - Pour Z_{100} :
     `[10.197  9.944  9.758  10.053  10.467  9.311  10.490  9.143  9.454  10.181]`

   Lequel semble être le meilleur estimateur ?

2. On suppose que les vecteurs `T` et `Z` contiennent maintenant chacun 10000 réalisations de T_{100} et de Z_{100}. On entre alors dans la console les commandes suivantes :

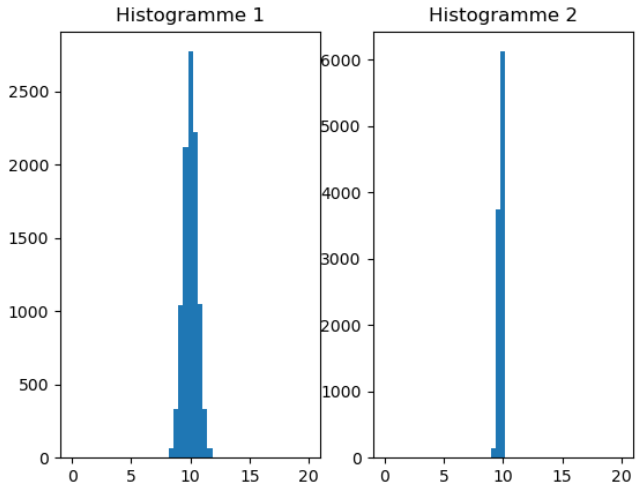
```
>>> np.mean(T)
9.696858
>>> np.mean(Z)
9.900282
>>> np.std(T)
0.194570
>>> np.std(Z)
0.447446
```

(a) A quoi correspond les valeurs obtenues ?

(b) Que peut-on dire des estimateurs T_{100} et Z_{100} ?

3. (a) Commander les histogrammes associées à chacun des deux vecteurs `T` et `Z`.

(b) Pour $a = 10$, on obtient les deux histogrammes suivants :



Attribuer chaque histogramme à un des estimateurs dont il représente la distribution. Justifier votre choix.

Exercice 7 (★★★)

On rappelle que la suite de Fibonacci (F_n) est la suite de nombre réels définis par :

$$\begin{cases} F_0 &= 0 \\ F_1 &= 1 \\ F_{n+2} &= F_{n+1} + F_n \end{cases}$$

Les premiers termes de la suite sont $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55...$

On admet que, pour tout entier naturel non nul n, il existe un unique entier k et un unique k-uplet d'entiers $(c_1, \dots, c_k)$ vérifiant :

- $c_1 \geq 2$,
- $c_i + 1 < c_{i+1}$,

tels que

$$n = \sum_{i=1}^{k} F_{c_i}.$$

Cette décomposition de n en somme de nombres de la suite de Fibonacci est appelée *décomposition de Zeckendorf* de n.

Par exemple, $17 = 13 + 3 + 1 = F_7 + F_4 + F_2$ donc $k = 3$ et $(c_1, c_2, c_3) = (2, 4, 7)$.

1. Écrire une fonction `def fibo(k):` qui renvoie le terme F_k de la suite (F_n).
2. Proposer un script qui permette d'obtenir la liste des 20 premier terme de la suite (F_n).
3. Écrire une fonction `def recherche(x, L):` qui prend en entrée :
   - un entier naturel `x` ;
   - une liste `L` déjà triée dans l'ordre croissant, dont le premier terme inférieur ou égal à `x` et le dernier est strictement supérieur à `x` ;

   et qui renvoie le plus grand élément de la liste `L` qui soit inférieur ou égal à `x`.
4. Que renvoie la fonction suivante ? Justifier votre réponse en détaillant étape par étape ce qu'elle fait.

```
def Zeckendorf(n):
    i = 0
    L = [fibo(i)]
    while L[-1]<=n:
        i = i+1
        L.append(fibo(i))
    k = n
    T = []
    while k>0:
        f = recherche(k, L)
        T.append(f)
        k = k-f
    return(T)
```

5. En quoi l'algorithme précédent est-il un algorithme glouton ?

Exercice 8 (★★)

1. Écrire une fonction `indicemin` en langage `Python` qui, pour une liste `L`, renvoie l'indice du minimum de la liste.
2. On considère la fonction :

```
def mystere(L):
    T = []
    n = len(L)
    for k in range(n):
        i = indicemin(L)
        T.append(L[i])
        del L[i]
    return(T)
```

   Que renvoie cette fonction `mystere` ? Justifier votre réponse en détaillant étape par étape ce qu'elle fait.

Exercice 9 (★)

1. On considère le script suivant :

```
1 A = np.array([[3, 4, 4], [2, -1, -2], [-2, 0, 1]])
2 L = al.eig(A)
3 print(L[0])
4 print(L[1])
```

   Que fait ce script ? Que font les lignes 3 et 4 ?

2. On exécute ce script et on obtient alors dans la console :

```
>>> [3.  1.  -1.]
>>> [[ 0.5773503   -2867D-16    -0.4082483]
     [ 0.5773503   -0.7071068    0.8164966]
     [-0.5773503    0.7071068   -0.4082483]]
```

   Déterminer à l'aide de ces résultats les valeurs propres de A et des vecteurs propres associés (chacun des vecteurs sera donné avec des coordonnées entières). A est-elle diagonalisable ?

Exercice 10 (★★)

On reprend ici les résultats obtenus dans l'**Exercice 15 - TD 1**.

1. On a démontré que l'équation $x^3 + x^2 + x - 1 = 0$ admet une unique solution $\alpha \in [1/3, 1]$.

   Écrire un programme basé sur l'algorithme de dichotomie pour obtenir une valeur approchée de α à une précision ε entrée par l'utilisateur.

2. On considère la suite (u_n) définie par :

$$u_0 = 1 \quad \text{et} \quad \forall n \in \mathbb{N},\ u_{n+1} = \frac{1}{u_n^2 + u_n + 1}.$$

   Écrire un programme qui, étant donné n, retourne la valeur de u_n.

3. On a démontré que :

$$\forall n \in \mathbb{N}, \quad |u_n - \alpha| \leq \left(\frac{135}{169}\right)^n.$$

   En déduire un programme pour obtenir une valeur approchée de α à une précision ε entrée par l'utilisateur.

Exercice 11 (★★)

On a démontré dans l'**Exercice 18 - TD 1** que, pour tout $n \in \mathbb{N}^*$, l'équation

$$x \ln(1 + x) = \frac{1}{n^2}$$

admet une unique solution strictement positive notée α_n.

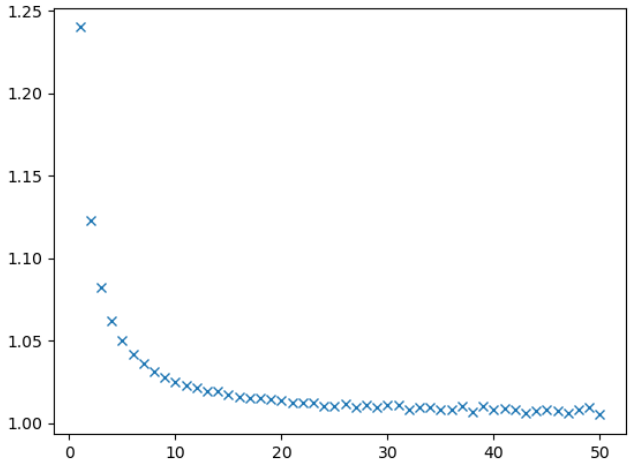
1. Compléter la fonction `Python` suivante pour que, étant donné $n \in \mathbb{N}^*$, elle retourne une approximation de α_n à 10^{-4} près.

```
1 def alpha(n):
2     x = np.arange(0, 2, 0.0001)
3     k = 0
4     while x[k]*np.log(1+x[k]) < 1/n**2:
5         .....
6     return( ..... )
```

2. A la suite du programme précédent, on ajoute les commandes suivantes :

```
12 n = np.arange(1, 51)
13 m = [k*alpha(k) for k in n]
14 plt.plot(n, m, 'x')
15 plt.show()
```

On obtient alors le résultat graphique suivant :

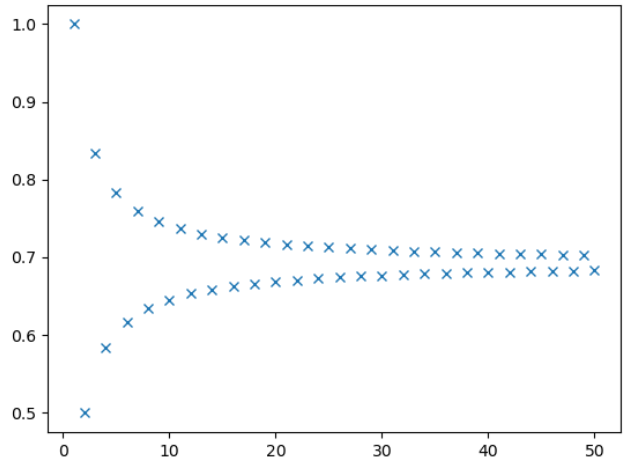


A quoi correspond la figure obtenue ? Que peut-on en déduire ?

Exercice 12 (★★)

On considère la série harmonique alternée $\sum_{n\geq 1} \frac{(-1)^{n-1}}{n}$. On note S_n sa n-ième somme partielle.

1. Écrire une commande `Python` 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 `Python` 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. Donner les commandes pour représenter graphiquement les points M_i de coordonnées $\left(i, \sum_{k=1}^{i} \frac{(-1)^{k-1}}{k}\right)$ pour $i = 1, \ldots, 50$.
4. On obtient le graphe suivant :



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

5. On admet que $\sum_{n=1}^{+\infty} \frac{(-1)^{n-1}}{n} = \ln(2)$. Écrire un script `Python` 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 13 (★★)

On considère les deux tables suivantes :

`Adhérent`

Id	Nom	Prénom	Age	Ville	Sport	Téléphone
1	Brun	Hugo	31	Agen	Judo	0664872435
2	Leroy	Léa	28	Nérac	Kung Fu	0687564129
3	Pic	Émile	29	Mézin	Aïkido	0675937765
4	Dulac	Noé	26	Dausse	Judo	0612532282

`Cotisation`

Sport	Cotis	Responsable
Judo	250	Henri F
Kung Fu	210	Michel T
Aïkido	320	Carole L

1. Identifier une clef primaire dans chacune des tables `Adhérent` et `Cotisation`, ainsi qu'une clef étrangère dans la table `Adhérent`.
2. Écrire une requête SQL pour sélectionner :
   (a) Les noms et prénoms des adhérents qui pratiquent le judo.
   (b) Le numéro de téléphone de M. Brun.
   (c) Le nom des adhérents qui ont moins de 30 ans.
   (d) Le numéro de téléphone des adhérents qui pratiquent le judo et habitent à Agen.
   (e) L'age des adhérents qui ne pratiquent pas le judo.
3. (a) Mme Leroy veut arrêter le Kung Fu et veut faire du karaté.
   Écrire une requête SQL permettant de modifier la table en ce sens.
   (b) Écrire une requête SQL augmentant tous les ages d'une unité.
4. (a) Écrire une requête SQL permettant d'obtenir les noms et prénoms des adhérents ayant une cotisation supérieure ou égale à 250 euros.
   (b) Écrire une requête SQL permettant de savoir qui est le responsable du sport pratiqué par Léa Leroy.
