Lycée Jacques Decour — Année 2025-2026

Devoir Surveillé III

Informatique

22 Novembre 2025 – Durée : 2 heures

L'usage de toute calculatrice, document extérieur, ou téléphone portable est strictement interdit. Un soin tout particulier devra être porté à la qualité et à la précision de la rédaction des argument.

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, il la signalera sur sa copie et poursuivra sa composition, motivant les initiatives qu'il sera amené à prendre.

Dans les exercices en `Python` de ce devoir, on supposera avoir importé les bibliothèques suivantes :

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import numpy.random as rd
```

On rappelle alors que l'exécution de `np.arange(a,b,1)` renvoie le `numpy-array` suivant : `[a,a+1,a+2,…,b-2,b-1]` (de a inclus à b exclus par pas de taille 1).

Exercice 1 Trois méthodes pour calculer $\sqrt{2}$

Dans cet exercice, on propose trois méthode pour calculer numériquement $\sqrt{2}$. Pour cela, on pose la fonction : $\left\{ \begin{array}{rcl} f : [1,2] & \to & \mathbb{R} \\ x & \mapsto & x^2 - 2 \end{array} \right.$

1– Recopier et compléter la fonction `Python` suivante pour implémenter en `Python` la fonction f donnée ci-dessus :

```
def f(x):
    return ____
```

2– **Méthode par balayage**

On considère la fonction suivante :

```
def balayage(p):
    u=np.arange(1,2,10**(-p))
    i=0
    while f(u[i])<0:
        i=i+1
    return u[i]
```

Expliquer ce que renvoie la commande `balayage(p)` ? On justifiera sa réponse en expliquant le fonctionnement de la fonction.

3– **Méthode par dichotomie**

Recopier et compléter la fonction `Python` suivante afin qu'elle renvoie une valeur approchée de $\sqrt{2}$ à ε près par la méthode de dichotomie :

```
def dicho(eps):
    a=1
    b=____
    c=____
    while ____:
        if ____:
            b=____
        else:
            ____
        c=____
    return c
```

4– **Méthode de Newton**

On considère la suite $(x_n)_{n\in\mathbb{N}}$ définie par $x_0 = 1$ et

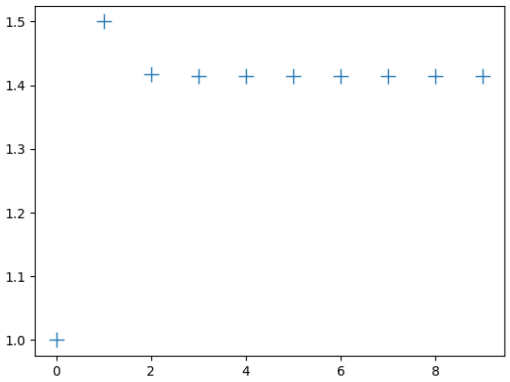
$$\forall n \in \mathbb{N},\ x_{n+1} = \frac{1}{2}\left(x_n + \frac{2}{x_n}\right)$$

a) Écrire une fonction `suite_x(n)` qui, étant donné l'argument n, renvoie la valeur de x_n correspondante.

b) Recopier et compléter la fonction suivante afin que, étant donné l'argument N, elle affiche une représentation graphique des N premiers termes de la suite :

```
def affiche_x(N):
    x = np.zeros(N)
    n = np.arange(0,N,1)
    for i in ____:
        x[i] = ____
    plt.figure()
    plt.plot(____,____,'+')
    plt.show()
```

c) On suppose que les fonctions précédentes on été codées puis exécutées correctement ; l'exécution de `affiche_x(12)` donne la figure ci-contre. On admet alors que $(x_n)_{n\in\mathbb{N}}$ converge vers une limite l.
Determiner cette limite l.



d) On admet que $\forall n \in \mathbb{N},\ |x_n - \sqrt{2}| \leqslant 3\left(\frac{1}{2}\right)^{2^n}$. Écrire une fonction `newton(eps)` qui, étant donné un réel `eps` $= \varepsilon > 0$, renvoie une valeur approchée de $\sqrt{2}$ à ε près.

Exercice 2 Une expérience d'urnes

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, et la variable aléatoire X_n grâce à `Python`.

1– Recopier et compléter la fonction `Python` suivante, qui simule le tirage d'une boule dans une urne contenant x boules rouges et y boules blanches et qui renvoie la valeur 0 si la boule est rouge et 1 si elle est blanche.

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

2– Recopier et compléter la fonction `Python` suivante, afin qu'elle simule n tirages successifs dans une urne contenant initialement a boules rouges et b boules blanches (selon le protocole décrit ci-dessus) et qui renvoie la valeur de X_n à la fin de l'expérience :

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

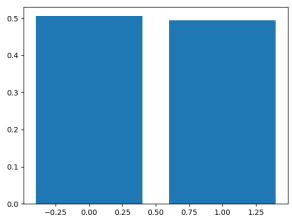
3– Justifier que $X_n(\Omega) = [\![0; n]\!]$.

4– Recopier et compléter la fonction `Python` suivante qui fait appel N fois à la fonction précédente pour estimer la loi de X_n par échantillonnage. Elle renverra une liste `L` contenant les approximations de $\mathbb{P}(X_n = 0), \mathbb{P}(X_n = 1), \ldots, \mathbb{P}(X_n = n)$.

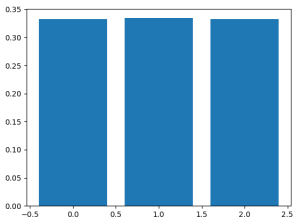
```
def simulation(a,b,n,N):
    L = np.zeros(n+1)
    for k in range(N):
        x = ____
        L[x] = ____
    return ____
```

5– On écrit la fonction `affiche_simulation(a,b,n)` suivante, qui nous permet de donner les graphiques suivants pour les cas $a = b = 1$ et différentes valeurs de n, sur un échantillon de taille 50000 :

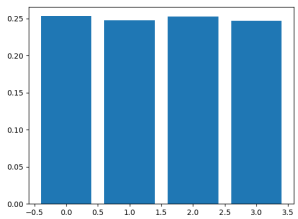
```
def affiche_simulation(a,b,n):
    plt.figure()
    k = np.arange(0,n+1,1)
    plt.bar(k, simulation(a,b,n,50000))
    plt.show()
```



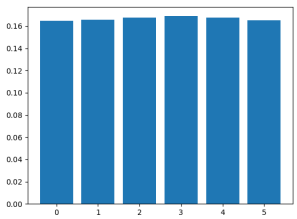
`affiche_simulation(1,1,1)`



`affiche_simulation(1,1,2)`



`affiche_simulation(1,1,3)`



`affiche_simulation(1,1,5)`

À l'aide de ces résultats, conjecturer sur la loi de X_n, en expliquant votre démarche.

Exercice 3 Propagation d'une rumeur dans un groupe

On considère un groupe de N personne ; chacune croit ou non une rumeur qui se propage dans ce groupe. On regarde chaque jour combien de membre du groupe croient cette rumeur, et on note ce nombre X_n. On considère qu'au jour 0, on a n_0 personnes qui croient la rumeur, donc $X_0 = n_0$. Chaque jour $n+1$, chaque membre du groupe revoit son avis, et croit la rumeur avec probabilité p_n, et ne la croit pas avec probabilité $1-p_n$, ou p_n est la proportion des membres qui croiyaient la rumeur la veille. On a donc $p_n = \dfrac{X_n}{N}$.

1– a) Soit $i \in [\![0; N]\!]$, et $k \in \mathbb{N}$. Reconnaître la loi conditionnelle de X_{k+1} sachant $[X_k = i]$. Donner alors l'image $X_{k+1}(\Omega)$ et l'expression de $\mathbb{P}_{[X_k=i]}(X_{k+1} = j)$ pour $j \in X_{k+1}(\Omega)$.

b) Recopier et compléter la fonction suivante pour que les composantes de la liste `L` renvoyée en sortie simulent les valeurs de $X_0, X_1, \ldots, X_k$, pour des valeurs k, N et n_0 données en entrées.

```
def simulX(k,N,n0):
    L=np.zeros(k+1)
    L[0]= ____
    for i in range(1,k+1):
        L[i]= ____
    return L
```

c) On exécute 4 fois la fonction `simulX` pour $k = 20$, $N = 10$, et différentes valeurs de n_0 :

– Pour $n_0 = 3$;

```
[3,2,3,2,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0]
[3,2,4,1,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0]
[3,5,5,7,7,8,9,7,8,8,7,8,10,10,10,10,10,10,10,10,10]
[3,3,1,3,2,2,2,3,3,2,1,2,2,1,3,2,2,4,1,2,1]
```

– Pour $n_0 = 5$;

```
[5,7,7,8,7,8,8,8,8,8,8,8,6,4,3,3,3,2,2,1,3]
[5,5,8,8,8,6,7,9,9,8,8,9,9,9,9,10,10,10,10,10,10]
[5,5,6,6,6,5,7,7,8,8,8,7,4,4,3,1,1,0,0,0,0]
[5,6,6,8,9,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10]
```

– Pour $n_0 = 8$;

```
[8,9,9,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10]
[8,8,6,7,8,8,7,7,7,8,6,5,2,2,1,1,1,0,0,0,0]
[8,6,6,6,7,9,9,8,7,7,8,9,7,5,4,5,3,4,2,1,2]
[8,7,8,9,9,8,8,6,9,10,10,10,10,10,10,10,10,10,10,10,10]
```

Commenter les résultats obtenus.

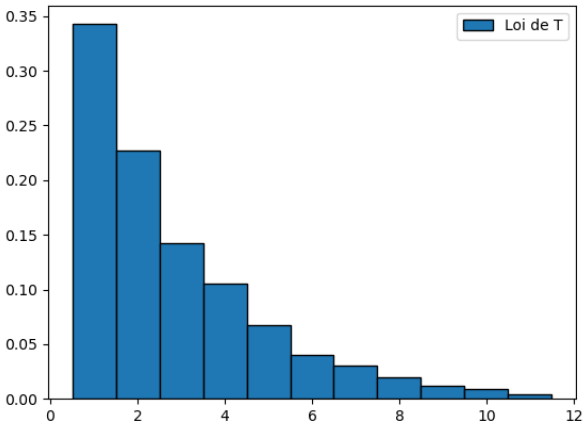
2– On pose dans cette question $N = 3$ et $n_0 = 1$. On s'intéresse à la variable aléatoire T égale au numéro du premier jour k tel que $X_k = 0$ ou $X_k = 3$.

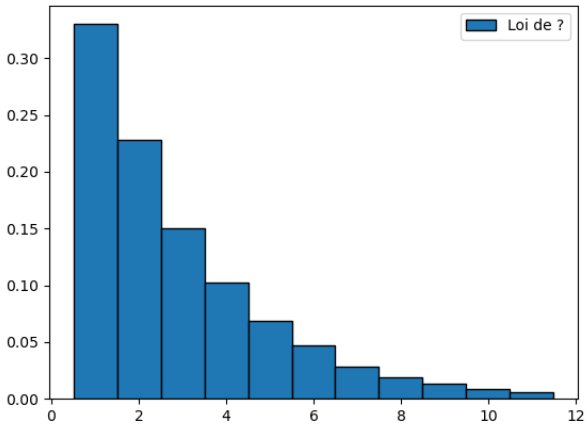
a) Recopier et compléter la fonction suivante afin qu'elle renvoie un vecteur `T` contenant 10000 réalisations de T.

```
def simulT():
    n0=1
    N=3
    s=10000
    T=np.zeros(s)
    for i in range(s):
        t=0
        X=n0
        while ___ and ____:
            t=t+1
            X= ____
        T[i]=t
    return T
```

b) On exécute alors les commandes suivantes, pour obtenir les deux graphes ci-dessous :

```
c = np.arange(0.5,12.5,1)
T=simulT()
G=rd.geometric(1/3,10000)
plt.figure()
plt.hist(T,c,density=True)
plt.figure()
plt.hist(G,c,density=True)
plt.show()
```





Que représentent ces deux graphiques ? Quelle conjecture pouvez-vous en tirer pour la loi de T ?

Exercice 4 Base de données 1

On dispose d'une base de données comportant deux tables, `vehicule` et `annonce`, décrites ci-dessous :

- La table `vehicule` recense des informations sur les modèles de véhicules en vente sur le marché. Elle est composée des attributs suivants :
  - `id_vehicule` (de type INTEGER) : un code permettant d'identifier de façon unique chaque référence de véhicule (marque et modèle).
  - `marque` (de type TEXT) : le nom du constructeur du véhicule.
  - `modele` (de type TEXT) : le modèle du véhicule, un constructeur proposant en général plusieurs modèles de véhicules à la vente.
  - `prix_neuf` (de type INTEGER) : prix de vente du véhicule neuf.
- La table `annonce` regroupe des informations sur un grand nombre d'annonces de véhicules d'occasion. Chaque enregistrement correspond à une annonce et possède les attributs suivants :
  - `id_annonce` (de type INTEGER) : un code permettant d'identifier chaque annonce de façon unique.
  - `id_vehicule` (de type INTEGER) : l'identifiant du modèle de véhicule vendu, qui correspond à l'identifiant utilisé dans la table vehicule.
  - `annee` (de type INTEGER) : année de première mise en circulation du véhicule.
  - `km` (de type INTEGER) : nombre de kilomètres parcourus par le véhicule au moment de la revente.
  - `prix_occasion` (de type INTEGER) : prix de vente du véhicule d'occasion.

1– En justifiant brièvement, identifier une clef primaire dans chacune des tables `vehicule` et `annonce`, ainsi qu'une clef étrangère dans la table `annonce`.

2– Donner le schéma relationnel associé aux tables `vehicule` et `annonce`.

3– Expliquer brièvement ce que la requête suivante renvoie :

```
SELECT COUNT(*) FROM vehicule WHERE prix_neuf > 100000;
```

4– Écrire une requête SQL permettant d'extraire les noms de tous les modèles de véhicules mis en vente par le constructeur (la marque) `Dubreuil Motors`.

5– Expliquer brièvement ce que la requête suivante renvoie :

```
SELECT id_annonce, id_vehicule FROM annonce
WHERE annee>=2018 AND km<300000 ORDER BY prix_occasion ASC ;
```

6– À l'aide d'une jointure, écrire une requête `SQL` permettant d'obtenir, sur une même table, la liste de toutes les annonces de la table `annonce` avec les attributs suivants :

- l'identifiant de l'annonce
- le kilométrage
- le prix de vente du véhicule neuf.
- le prix de l'annonce d'occasion

7– Écrire une requête `SQL` qui renvoie la marque, le modèle, et l'écart de prix entre le prix neuf et le prix d'occasion, pour chaque annonce. On triera les résultats par ordre décroissants d'écart de prix.