Thierry Gaspari

L'ORAL DE MATHS ET INFORMATIQUE AUX CONCOURS AGRO-VÉTO ET G2E

BCPST

j'intègre

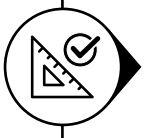
ANNALES ET SUJETS ORIGINAUX CORRIGÉS

DUNOD

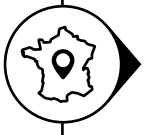
Couverture : création Hokus Pokus, adaptation Studio Dunod

NOUS NOUS ENGAGEONS EN FAVEUR DE L'ENVIRONNEMENT :

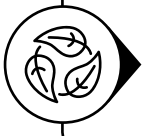
Nos livres sont imprimés sur des papiers certifiés pour réduire notre impact sur l'environnement.



Le format de nos ouvrages est pensé afin d'optimiser l'utilisation du papier.



Depuis plus de 30 ans, nous imprimons 70% de nos livres en France et 25% en Europe et nous mettons tout en œuvre pour augmenter cet engagement auprès des imprimeurs français.



Nous limitons l'utilisation du plastique sur nos ouvrages (film sur les couvertures et les livres).

11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-086701-1

Avant-propos

Cet ouvrage propose des **exercices corrigés de préparation aux oraux des concours Agro-Véto et G2E**. Il s'adresse aux étudiants en seconde année de BCPST qui souhaitent **réussir l'écrit et l'oral de ces deux concours**. Les exercices sont des **sujets donnés aux concours**, ou bien des **sujets type** conçus conformément à la réforme des concours effective depuis 2023, en accord avec le nouveau programme 2022 des classes préparatoires BCPST.
Chaque exercice bénéficie d'une **correction complète avec des explications détaillées**, des rappels de cours, une variété de méthodes, des conseils tactiques et des explications sur les techniques de calcul. La progressivité de la difficulté des questions, ainsi que la profusion des explications dans les corrections, permettront à tout étudiant de réaliser des progrès notables. **Les** 170 **programmes informatiques sont suivis d'explications détaillées.** De plus, certains approfondissements proposés en fin d'exercices, signalés par le pictogramme  , sont accessibles sur le site www.dunod.com, sur la page de l'ouvrage.
Pour les étudiants peu à l'aise en Mathématiques ou en Informatique, les explications fournies constitueront une ressource précieuse pour gagner en confiance et progresser. Les étudiants ayant pour unique objectif l'un des deux concours trouveront un intérêt évident à étudier les sujets de l'autre concours, car les thèmes abordés sont similaires, de même que les méthodes de résolution. Les questions de fin d'exercices sont à un niveau élevé, permettant ainsi aux étudiants les plus ambitieux de se confronter à des raisonnements complexes, en les préparant ainsi au mieux aux concours à venir.

Quand et comment utiliser cet ouvrage ?

Le livre peut être utilisé **dès le début de la seconde année**, afin de **se préparer aux khôlles et aux évaluations écrites**. Les **tableaux récapitulatifs** donnés au début des chapitres 2 et 4 permettront à l'étudiant de savoir quels exercices il peut aborder en fonction des chapitres étudiés en cours d'année. L'ouvrage peut aussi être utilisé **au moment de la préparation des oraux**, afin d'élever son niveau juste avant l'épreuve.
Dans les corrections, l'analyse des questions et les remarques méthodologiques sont séparées de la rédaction précise, qui est signalée par le pictogramme  .

Les explications détaillées des programmes informatiques sont signalées par  .

Présentation détaillée du livre

Cet ouvrage est séparé en deux parties :

La première partie concerne l'oral MPI (Mathématiques Pratiques et Informatique) du concours Agro-Véto, structuré en trois exercices distincts :

— L'oral commence par une **question de cours**. **Le chapitre** 1 **comporte les** 121 **questions posées au concours** 2023**, ainsi que toutes les réponses**. Le rapport du jury du concours 2023 indique que « *les questions de cours ne sont réussies qu'à* 70 % » et que « *l'effet n'est pas très positif quand la question*

de cours est imprécise ou bâclée ». Ce chapitre 1 permettra aux étudiants d'être parfaitement préparés pour cette première partie de l'oral.

— Le deuxième moment consiste à **présenter l'exercice préparé**, qui est un exercice de Mathématiques contenant quelques questions d'informatique, l'étudiant disposant d'un ordinateur sur lequel il peut écrire ses programmes en langage Python. **Dans le chapitre** 2 **on présente** 20 **sujets complets, entièrement corrigés**. Certains ont été donnés au concours 2023, les sujets antérieurs à 2023 ont été aménagés de manière à être réalisés sur 40 minutes. **Le chapitre** 2 **débute par un tableau récapitulatif**, donnant pour chaque exercice sa difficulté et les thèmes abordés.

— Enfin, **la troisième séquence est un court exercice d'informatique donné sans préparation**. **Dans le chapitre** 3 **on présente** 15 **sujets, corrigés de façon très détaillée**. Ici encore tous les programmes informatiques sont expliqués avec beaucoup de soin et de pédagogie, pour permettre à l'étudiant de comprendre et progresser.

La seconde partie de l'ouvrage est consacrée à l'oral de Mathématiques et à l'oral d'informatique du concours G2E.

— **Dans le chapitre** 4 **on présente** 20 **exercices corrigés de l'oral de Mathématiques.** Lors de cet oral l'étudiant prépare puis présente deux exercices : l'un sur l'Analyse ou l'Algèbre, l'autre sur les Probabilités. **En début de chapitre** 4 **l'étudiant trouvera un tableau récapitulatif avec la difficulté de chaque exercice et les notions abordées**.

— **Le chapitre** 5 **est consacré à l'oral d'Informatique du concours G2E**. L'étudiant prépare puis présente un premier exercice, le jury lui donne ensuite un second exercice de même format à traiter sans préparation. **Cet ouvrage propose** 10 **sujets d'informatique complets**, corrigés avec beaucoup de détails et d'explications.

En conclusion

L'étudiant trouvera dans cet ouvrage des exercices classiques qui abordent l'ensemble des notions de Mathématiques et d'Informatique étudiées pendant les deux années de BCPST ; des explications méthodologiques et tactiques pour savoir comment gérer l'oral ; des corrections entièrement rédigées avec des explications sur le cours et les techniques de calcul ; 170 programmes Python détaillés et totalement expliqués.

Au final cet ouvrage est **bien plus qu'un recueil d'exercices corrigés** :
il permettra de mieux comprendre les notions abordées en Mathématiques et en Informatique pendant les deux années de classes préparatoires. Il aidera les étudiants qui en font l'acquisition dès le début de seconde année à préparer méthodiquement leurs khôlles. Il améliorera la compréhension des méthodes et des concepts mathématiques indispensables à l'écrit des concours Agro-Véto et G2E. Enfin il permettra des révisions efficaces en fin de seconde année, pour réussir les oraux.

Table des matières

Oraux du concours Agro-Véto

Oraux du concours G2E

Partie 1

Oraux du concours Agro-Véto

Oraux du concours Agro-Véto

CHAPITRE 1

Questions de cours

Toutes ces questions sont issues du concours Agro-Véto 2023.

Exercice 1.1 : Injectivité

Définition d'une application $f : E \to G$ injective.

$f : E \to G$ est injective si et seulement si tout élément de G a au plus un antécédent par f dans E.
Autre formulation : $f : E \to G$ est injective si et seulement si, pour tout y de G, l'équation $f(x) = y$ admet au maximum une solution x dans E.

Exercice 1.2 : Surjectivité

Définition d'une application $f : E \to G$ surjective.

$f : E \to G$ est surjective si et seulement si tout élément de G a au moins un antécédent par f dans E.
Autre formulation : $f : E \to G$ est surjective si et seulement si, pour tout y de G, l'équation $f(x) = y$ admet au moins une solution x dans E.

Exercice 1.3 : Bijection réciproque

Pour $f : E \to F$ bijective, définir la fonction réciproque f^{-1}.

$f^{-1} : F \to E, y \mapsto f^{-1}(y) =$ l'unique antécédent de y par f dans E.
Autrement dit,

$$x = f^{-1}(y) \Leftrightarrow f(x) = y.$$

Exercice 1.4 : Famille génératrice

Définition d'une famille génératrice dans un espace vectoriel E.

Une famille de vecteurs $(u_1, \ldots, u_q)$ est génératrice de E si et seulement si tout vecteur de E est une combinaison linéaire des vecteurs $u_1, \ldots, u_q$.

Exercice 1.5 : Liberté

Définition d'une famille libre de vecteurs dans un espace vectoriel E.

La famille $(u_1, \ldots, u_k)$ est libre si et seulement si la seule combinaison linéaire de cette famille qui soit nulle est la combinaison linéaire dont tous les coefficients sont nuls.

- *Autre formulation* :
La famille $(u_1, \ldots, u_k)$ est libre si et seulement si aucun vecteur de cette famille ne peut s'écrire comme combinaison linéaire des autres vecteurs de la famille.

- *Autre formulation* :
La famille $(u_1, \ldots, u_k)$ est libre si et seulement si :

$$\forall (\lambda_1, \ldots, \lambda_k) \in \mathbb{K}^k, \ \sum_{i=1}^{k} \lambda_i u_i = 0_E \Rightarrow \forall i \in [\![1, k]\!], \ \lambda_i = 0.$$

Exercice 1.6 : Base

Donner la définition d'une base d'un espace vectoriel.

La famille de vecteurs $(u_1, \ldots, u_n)$ est une base de E si et seulement si $(u_1, \ldots, u_n)$ est libre et génératrice de E.

CHAPITRE 2

Exercices avec préparation

Exercice	Difficulté de 1 à 5	Fonctions et suites 1	Intégrales 1	Équations différentielles 1	Fonctions de plusieurs variables 1	Variables finies 1	Algèbre linéaire 1	Séries 2	Intégrales généralisées 2	Variables discrètes 2	Variables à densité 2	Polynômes 2	Diagonalisation 2	Produit scalaire 2
2.1	♠	✓					✓					✓	✓	
2.2	♠♠	✓				✓								
2.3	♠♠			✓			✓						✓	
2.4	♠♠	✓						✓		✓				
2.5	♠♠				✓								✓	✓
2.6	♠♠					✓	✓						✓	
2.7	♠♠♠	✓	✓						✓		✓			
2.8	♠♠♠	✓				✓				✓				
2.9	♠♠♠	✓	✓					✓	✓	✓				
2.10	♠♠♠	✓				✓								
2.11	♠♠♠♠	✓					✓						✓	
2.12	♠♠♠♠				✓		✓							✓
2.13	♠♠♠♠	✓	✓					✓	✓	✓	✓			
2.14	♠♠♠♠			✓			✓							
2.15	♠♠♠♠		✓			✓			✓		✓			
2.16	♠♠♠♠♠						✓					✓	✓	
2.17	♠♠♠♠♠	✓				✓								
2.18	♠♠♠♠♠	✓	✓					✓	✓	✓	✓			
2.19	♠♠♠♠♠	✓	✓	✓					✓		✓			
2.20	♠♠♠♠♠	✓				✓								

Lecture du tableau : la difficulté d'un exercice varie entre 1 et 5, le chiffre écrit après le nom du chapitre (1 ou 2) précise s'il est étudié en BCPST1 ou en BCPST2.

Exercice 2.1 : Diagonalisation et équations matricielles

On va étudier l'équation matricielle d'inconnue $A \in \mathscr{M}_{3,3}(\mathbb{R})$:

$$(*) \quad A^3 - A^2 - 8A + 11I_3 = 0.$$

On définit le polynôme P par : $P(X) = X^3 - X^2 - 8X + 11$.

1. *Racines de P.*

a. Démontrer qu'un polynôme de degré 3 à coefficients réels admet toujours au moins une racine dans $\mathbb{R}$.

b. A l'aide d'une représentation graphique obtenue avec le logiciel Python, conjecturer le nombre de racines de P ainsi que leurs valeurs.

c. En calculant les valeurs de P en 0 et 2, ainsi que ses limites en $-\infty$ et $+\infty$, justifier rigoureusement le nombre de racines de P.

2. *Étude de l'équation* $(*)$.

a. Déterminer les matrices diagonales solutions de $(*)$. Déterminer le nombre de solutions de ce type.

b. Déterminer les matrices diagonalisables solutions de $(*)$.

c. Soit $A = \begin{pmatrix} 2 & -1 & 1 \\ -1 & 0 & 2 \\ 1 & 2 & -1 \end{pmatrix}$. Montrer que A est solution de $(*)$.

3. On essaie maintenant de construire une méthode numérique permettant d'approcher une racine de P. On pose $g : \mathbb{R} \to \mathbb{R}, t \mapsto g(t) = P(t) + t$.

Écrire une fonction Python qui prend en entrée un réel a et un entier N, puis qui renvoie la liste $[u_0, u_1, \ldots, u_N]$, où (u_n) est la suite récurrente définie par un premier terme $u_0 = a$ puis par la relation de récurrence :

$$\forall n \in \mathbb{N}, \quad u_{n+1} = g(u_n).$$

Expliquer pourquoi l'éventuelle limite de (u_n) est une racine de P.

Tester numériquement. Cette méthode est-elle efficace ?

Analyse du sujet et commentaires tactiques :

Ce sujet est centré sur l'algèbre linéaire : calcul matriciel et diagonalisation. Il comporte aussi des questions sur les polynômes et l'analyse de première année. Pour les questions d'analyse l'approche graphique, réalisée grâce au logiciel Python, aide beaucoup, il faut donc connaître les commandes pour tracer une courbe rapidement. Les questions d'algèbre ne sont pas très difficiles si on comprend le lien avec le début du sujet. La dernière question d'informatique est sur les suites récurrentes, c'est très classique, l'étudiant devra prendre du recul pour interpréter les résultats obtenus.

1. **a**. Lorsqu'on veut justifier l'existence d'une solution à une équation, sans être en capacité de la résoudre, on pense au **théorème des valeurs intermédiaires**.

Rappel de cours :

Le **théorème des valeurs intermédiaires** énonce que si $a < b$ et f est une fonction continue sur $[a, b]$, alors pour tout y entre $f(a)$ et $f(b)$ il existe c dans $[a, b]$ tel que $f(c) = y$.

Soit P un polynôme de degré 3. Il existe donc (a_0, a_1, a_2, a_3) dans $\mathbb{R}^4$, avec $a_3 \neq 0$, tel que :

$$\forall t \in \mathbb{R}, \quad P(t) = a_3t^3 + a_2t^2 + a_1t + a_0.$$

Supposons pour l'instant $a_3 > 0$. Alors

$$\lim_{t\to+\infty} P(t) = \lim_{t\to+\infty} a_3t^3 = +\infty$$

et

$$\lim_{t\to-\infty} P(t) = \lim_{t\to-\infty} a_3t^3 = -\infty.$$

De plus P est une fonction continue sur $\mathbb{R}$. On peut donc appliquer le théorème des valeurs intermédiaires pour en déduire que P a au moins une racine réelle.
Maintenant, si $a_3 < 0$ les deux limites sont inversées mais la conclusion reste identique.

b.

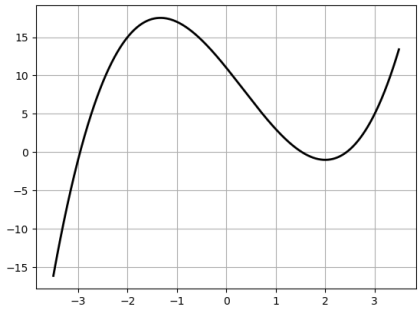
On définit la fonction $x \mapsto P(x)$, puis on trace P sur $[-3.5; 3.5]$, intervalle choisi après plusieurs essais.

```
import numpy as np
import matplotlib.pyplot as plt
def P(x):
    return(x**3-x**2-8*x+11)

def courbe():
    Lx=np.arange(-3.5,3.5,0.01)
    Ly=[P(t) for t in Lx]
    plt.plot(Lx,Ly)
    plt.grid()
    plt.show()
```

Explication détaillée du programme informatique :
Dans la fonction `courbe()`, on crée la liste `Lx` qui contient une suite arithmétique de réels allant de -3.5 à 3.49 avec un pas de 10^{-2}.
On utilise pour cela la commande `np.arange(a,b,h)`, importée du module `numpy`, qui crée une liste régulièrement échantillonnée (c'est-à-dire arithmétique) allant de `a` jusqu'à `b` (exclus) avec un pas de `h`.
Puis on construit la liste `Ly` qui contient les images de tous les points de `Lx` par la fonction P. Enfin on trace la courbe demandée avec `plt.plot()`, en affichant une grille avec la commande `plt.grid()`, ces commandes graphiques provenant du module `matplotlib.pyplot`.

Est obtenu ceci :



Il semble que P possède trois racines réelles, ayant pour valeurs approximatives -3, 1.5 et 2.4.

c.

Tout d'abord :

$$\lim_{x\to-\infty} P(x) = \lim_{x\to-\infty} x^3 = -\infty.$$

Puis $P(0) = 11 > 0$, et $P(2) = -1 < 0$. Enfin,

$$\lim_{x\to+\infty} P(x) = \lim_{x\to+\infty} x^3 = +\infty.$$

On peut alors appliquer le **théorème des valeurs intermédiaires** successivement sur les intervalles $]-\infty, 0]$, puis sur $]0, 2]$ et enfin sur $]2, +\infty[$, pour affirmer que P a au moins une racine dans chacun de ces trois intervalles. De plus P est de degré 3 donc P ne peut pas avoir plus de trois racines. En conclusion, P a exactement trois racines réelles que nous appelerons α, β, γ, avec

$$\alpha < 0 < \beta < 2 < \gamma.$$

2. **a**. Pour cette question simple, ne confondez pas « diagonale » avec « diagonalisable » !

Soit C une matrice diagonale. Il existe trois réels a_1, a_2, a_3 tels que :

$$C = \mathrm{diag}(a_1, a_2, a_3) = \begin{pmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{pmatrix}.$$

Les propriétés des matrices diagonales permettent d'affirmer que :

$$\begin{aligned} P(C) &= C^3 - C^2 - 8C + 11I_3 \\ &= \begin{pmatrix} a_1^3 - a_1^2 - 8a_1 + 11 & 0 & 0 \\ 0 & a_2^3 - a_2^2 - 8a_2 + 11 & 0 \\ 0 & 0 & a_3^3 - a_3^2 - 8a_3 + 11 \end{pmatrix}. \end{aligned}$$

C'est le moment décisif, où il faut être concentré et reconnaître le polynôme P !

On réécrit simplement :

$$C^3 - C^2 - 8C + 11I_3 = \begin{pmatrix} P(a_1) & 0 & 0 \\ 0 & P(a_2) & 0 \\ 0 & 0 & P(a_3) \end{pmatrix}.$$

Ainsi :

$$C \text{ est solution de } (*) \Leftrightarrow P(a_1) = P(a_2) = P(a_3) = 0.$$

Par conséquent, les matrices diagonales solutions de $(*)$ sont les matrices qui s'écrivent $\text{diag}(a_1, a_2, a_3)$ avec, pour tout i de $[\![1, 3]\!]$, $a_i \in \{\alpha, \beta, \gamma\}$.
Cet ensemble de solutions diagonales est constitué de $3^3 = 27$ matrices différentes.

b.

Soit A une matrice diagonalisable. Il existe une matrice P inversible et une matrice D diagonale telles que :

$$A = PDP^{-1}.$$

Les calculs de puissances successifs donnent alors :

$$A^2 = PD^2P^{-1} \quad ; \quad A^3 = PD^3P^{-1}.$$

et ainsi, en remarquant que $I_3 = PI_3P^{-1}$:

$$A^3 - A^2 - 8A + 11I_3 = P\big(D^3 - D^2 - 8D + 11I_3\big)P^{-1}.$$

On raisonne maintenant par équivalence :

$$\begin{aligned} A \text{ est solution de } (*) &\Leftrightarrow P\big(D^3 - D^2 - 8D + 11I_3\big)P^{-1} = 0 \\ &\Leftrightarrow D^3 - D^2 - 8D + 11I_3 = 0 \\ &\qquad (\text{car } P \text{ et } P^{-1} \text{ sont inversibles}) \\ &\Leftrightarrow \exists (a,b,c) \in \{\alpha,\beta,\gamma\}^3, \ D = \text{diag}(a,b,c). \end{aligned}$$

Il ne faut pas chercher à être plus précis car l'équation $(*)$ ne donne aucune information sur la matrice P. En conclusion, les matrices diagonalisables solutions de $(*)$ sont les matrices diagonalisables dont le spectre est inclus dans l'ensemble $\{\alpha, \beta, \gamma\}$.

c. Les calculs demandés ici peuvent être sources d'erreurs, l'étudiant peut pendant sa préparation utiliser le logiciel Python pour les « vérifier »(en fait, pour les effectuer). Il suffit d'écrire

$$\texttt{A} = \texttt{np.matrix}([[2, -1, 1], [-1, 0, 2], [1, 2, -1]])$$

puis de demander `A**2` ou `A**3`.

On calcule successivement les puissances de A :

$$A^2 = \begin{pmatrix} 6 & 0 & -1 \\ 0 & 5 & -3 \\ -1 & -3 & 6 \end{pmatrix} \quad ; \quad A^3 = \begin{pmatrix} 11 & -8 & 7 \\ -8 & -6 & 13 \\ 7 & 13 & -13 \end{pmatrix}.$$

puis on calcule $A^3 - A^2 - 8A + 11I_3$, qui vaut effectivement la matrice nulle. Donc A est solution de $(*)$.

Attention, une mauvaise idée ici aurait été d'étudier la diagonalisabilité de A en cherchant ses valeurs propres. C'est possible, mais les calculs sont compliqués.

3. L'idée du programme présenté est assez naturelle : pour résoudre $f(x) = 0$, on s'intéresse à la suite récurrente $u_{n+1} = g(u_n)$ avec $g(t) = f(t) + t$. Si cette suite converge alors on peut prouver que sa limite l est un point fixe de g, donc vérifie $f(l) = 0$.

Avant de programmer, étudions l'éventuelle limite de (u_n) en supposant que la suite (u_n) converge vers un réel L.
On sait que pour tout entier n, $u_{n+1} = P(u_n) + u_n$.
Puisque P est une fonction continue, on peut passer à la limite quand n tend vers $+\infty$ pour obtenir : $L = P(L) + L$, donc $P(L) = 0$ et ainsi $L \in \{\alpha, \beta, \gamma\}$.
On programme maintenant la suite récurrente :

```
def suite(a,N):
    u=a
    L=[u]
    for n in range(N):
        u=P(u)+u
        L=L+[u]
    return(L)
```

Explication détaillée du programme informatique :
On initialise le réel u égal à a qui est le premier terme de la suite, et on crée la liste L contenant au début ce premier terme. Ensuite, dans la boucle de répétition `for`, on écrit `u=P(u)+u` pour traduire la formule de récurrence $u_{n+1} = P(u_n) + u_n$, et on ajoute ce nouveau terme de la suite dans la liste L. Une fois sorti de la boucle on retourne la liste $L = [u_0, u_1, \ldots, u_N]$.

Les tests sont malheureusement décevants, par exemple `suite(1,4)` donne :

$$[1, 4, 31, 28624, 23451779020891]$$

et il en est de même pour les autres tests : la suite (u_n) ne converge pas et diverge très rapidement vers l'infini. Le candidat pourrait proposer comme amélioration de travailler avec $g(x) = \dfrac{P(x)}{100} + x$ (c'est un exemple), qui permet de diminuer les pentes de la fonction et de rendre plus probable la convergence de la suite récurrente. Ou bien on aurait pu penser à un algorithme de dichotomie.

CHAPITRE 3

Exercices sans préparation

Exercice 3.1 : Séquences d'ADN [Agro-Véto 2023]

1. Écrire une fonction `gene(n)` qui prend en entrée un entier naturel non nul n et retourne une chaîne de `n` caractères formée aléatoirement, de façon équiprobable, des caractères `'A','C','G','T'`.
2. Écrire une fonction `nbAC(g)` qui prend en entrée une chaîne de caractères formée des caractères `'A','C','G','T'` et qui retourne le nombre de fois où la séquence `'AC'` est présente.
Par exemple, `nbAC('GAGCACCCTCACTAA')` retournera 2.

Analyse du sujet et commentaires tactiques :
Ce sujet, très court et simple, porte sur les chaînes de caractères, qui se manipulent comme les listes. Pour la question 1 il faut créer une répétition de n tirages aléatoires, on utilisera les fonctions du module `random`. Pour la question 2 il suffit de créer un compteur en effectuant un parcours de la chaîne donnée en entrée.

1.

Le plus simple est de créer, au début, l'alphabet `'ACGT'` puis de piocher au hasard n fois dedans.

```
import random as rd
def gene(n):
    Alphabet='ACGT'
    chaine=''
    for k in range(n):
        indice=rd.randint(0,3)
        chaine=chaine+Alphabet[indice]
    return(chaine)
```

Explication détaillée du programme informatique :
On initialise `chaine` sous forme de chaîne de caractères vide. Dans la **boucle de répétition** `for`, on répète n fois les opérations suivantes :
- on tire un entier `indice` aléatoire entre 0 et 3 ;

- la lettre `Alphabet[indice]` est ajoutée à `chaine`.
Quand la boucle `for` est terminée, le programme retourne `chaine`.

On teste, par exemple : `gene(10)` nous donne : `'TGCTTCATAA'`

2.

On crée un **compteur** S :

```
def nbAC(g):
    S=0
    n=len(g)
    for k in range(n-1):
        if g[k]=='A' and g[k+1]=='C':
            S+=1
    return(S)
```

Explication détaillée du programme informatique :
On initialise `S` à 0, on calcule `n`, le nombre de lettres dans la chaîne de caractères `g`.
Ensuite on parcourt les indices `k` de 0 à $n-2$ et on teste si `'g[k]g[k+1]'='AC'` et, si c'est le cas, `S` augmente de 1. Il faut s'arrêter avec $k = n-2$ pour éviter l'erreur `out of range`, car `g[n-1]` est le dernier terme de la chaîne `g`.
Une fois la boucle `for` terminée, le programme retourne le compteur `S`.

Quelques tests montrent que le programme fonctionne comme prévu. Puisque l'exercice est court, le candidat qui le réussit peut s'attendre à quelques questions d'approfondissement, comme celles que l'on peut trouver dans cet ouvrage, dans l'exercice 5.5 d'oral de G2E sur les séquences d'ADN.

Exercice 3.2 : Numéros les plus fréquents [Agro-Véto 2023]

On s'intéresse dans cet exercice à des listes d'entiers. L'utilisation de `max` ou de `count` est interdite.
1. Écrire une fonction prenant en argument une liste d'entiers `L` et un entier $k \in \mathbb{N}$. Cette fonction renvoie `True` si tous les entiers de cette liste sont compris entre 0 et k et `False` sinon. Par exemple, pour `L` $= [0, 2, 0]$, la fonction renvoie `True` si $k = 2$ ou $k = 3$ et `False` si $k = 1$.
2. Écrire une fonction de mêmes arguments, où la liste est supposée ne contenir que des entiers compris entre 0 et k. Cette fonction renvoie l'élément le plus fréquent (ou l'un d'entre eux s'il y en a plusieurs).
Par exemple, pour `L` $= [0, 4, 0, 1, 4]$ et $k = 5$, la fonction renvoie 0 ou 4.

Partie 2

Oraux du concours G2E

Oraux du concours G2E

CHAPITRE

4 Oraux de Mathématiques

Exercice	Difficulté de 1 à 5	Fonctions et suites 1	Intégrales 1	Équations différentielles 1	Fonctions de plusieurs variables 1	Variables finies 1	Algèbre linéaire 1	Séries 2	Intégrales généralisées 2	Variables discrètes 2	Variables à densité 2	Polynômes 2	Diagonalisation 2	Produit scalaire 2
4.1	♠						✓					✓	✓	
4.2	♠	✓	✓											
4.3	♠						✓						✓	
4.4	♠♠								✓		✓			
4.5	♠♠	✓				✓		✓		✓				
4.6	♠♠	✓			✓									
4.7	♠♠	✓	✓			✓					✓			
4.8	♠♠					✓								
4.9	♠♠	✓												
4.10	♠♠♠						✓							
4.11	♠♠♠					✓								
4.12	♠♠♠	✓						✓						
4.13	♠♠♠♠								✓	✓	✓			
4.14	♠♠♠♠						✓						✓	✓
4.15	♠♠♠♠							✓		✓				
4.16	♠♠♠♠♠	✓				✓			✓		✓			
4.17	♠♠♠♠♠												✓	✓
4.18	♠♠♠♠♠					✓								
4.19	♠♠♠♠♠	✓	✓						✓		✓			
4.20	♠♠♠♠♠	✓		✓										

Lecture du tableau : la difficulté d'un exercice varie entre 1 et 5, le chiffre écrit après le nom du chapitre (1 ou 2) précise s'il est étudié en BCPST1 ou en BCPST2.

Exercice 4.1 : Espace vectoriel de polynômes [G2E 2021]

On note E l'espace vectoriel $\mathbb{R}_2[X]$ et on considère l'application Ψ définie sur E par :

$$\forall P \in E, \quad \Psi(P) = X^2 P\big(\frac{1}{X}\big).$$

1. Montrer que Ψ est un endomorphisme de E.
2. Donner la matrice A de Ψ dans la base canonique de E.
3. Justifier que A est diagonalisable et montrer que $\Psi^2 = Id_E$.
4. Trouver les valeurs propres et sous-espaces propres de Ψ.

Analyse du sujet et commentaires tactiques :
On étudie ici une application linéaire dans un espace vectoriel de polynômes. Pour les questions 1 et 2 le plus simple est de calculer l'image du polynôme $R_0 = 1$ par Ψ, puis l'image de $R_1 = X$, puis l'image de X^2. On aura alors obtenu la matrice A. Les calculs matriciels permettront alors de continuer l'exercice. La recherche des valeurs propres peut être abordée de plusieurs façons, mais ici la tactique suggérée par la construction de l'énoncé est d'utiliser la relation polynômiale vérifiée par Ψ et de prouver que les valeurs propres satisfont la même équation.

1. Rappel de cours :
Ψ est un **endomorphisme** de E si, et seulement si, Ψ est une application linéaire et $\Psi(E) \subset E$.

Considérons deux éléments P et Q de E ainsi que deux réels a et b.

$$\Psi(aP + bQ) = X^2\Big(aP\big(\frac{1}{X}\big) + bQ\big(\frac{1}{X}\big)\Big) = a\Psi(P) + b\Psi(Q)$$

donc Ψ est une **application linéaire**.
Calculons les images par Ψ des vecteurs de la base canonique.

$$\Psi(1) = X^2 \quad ; \quad \Psi(X) = X \quad ; \quad \Psi(X^2) = 1.$$

Toutes ces images sont dans E, donc Ψ **va de** E **dans** E.
Ainsi Ψ est un **endomorphisme** de E.

2. Rappel de cours :
La matrice de Ψ dans la base canonique $\mathscr{B} = (1, X, X^2)$ s'obtient en écrivant dans la $(k+1)$-ième colonne les coordonnées dans la base $\mathscr{B}$ du vecteur $\Psi(X^k)$.

En utilisant les calculs de la question précédente :

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

3.

A est **symétrique à coefficients réels donc diagonalisable**.
Par calcul matriciel on obtient $A^2 = I_3$, donc $\Psi^2 = Id_E$.

4. Méthode : la méthode classique qui consiste à calculer le rang de $(A - \lambda I_3)$ par pivot de Gauss est possible et techniquement facile ici. Mais on va plutôt suivre l'idée de l'énoncé en démontrant que les valeurs propres de Ψ sont solutions de l'équation $\lambda^2 = 1$.

Soit λ une valeur propre de Ψ. Il existe un vecteur Q non nul tel que $\Psi(Q) = \lambda Q$.
Puis

$$\Psi^2(Q) = \Psi\big(\Psi(Q)\big) = \Psi(\lambda Q) = \lambda\Psi(Q) = \lambda^2 Q.$$

De plus $\Psi^2 = Id_E$, donc $\lambda^2 Q = Q$ et ainsi

$$(\lambda^2 - 1)Q = 0.$$

Puisque le vecteur Q est non nul, on a : $\lambda^2 = 1$.
Ainsi les valeurs propres de Ψ sont contenues dans $\{-1, 1\}$.
- Etude de $E_1 = \mathrm{Ker}(\Psi - Id_E)$: on écrit la matrice

$$A - I_3 = \begin{pmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{pmatrix}$$

qui permet d'obtenir

$$E_1 = Ker(\Psi - Id_E) = Vect(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}) = Vect(X, 1 + X^2).$$

- Etude de $E_{-1} = \mathrm{Ker}(\Psi + Id_E)$: ici la matrice est

$$A + I_3 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

donc

$$E_{-1} = Ker(\Psi + Id_E) = Vect(\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}) = Vect(1 - X^2).$$

En conclusion, le spectre de Ψ est $\{-1, 1\}$.
On retrouve bien le fait que Ψ est diagonalisable puisque

$$\dim(E_1) + \dim(E_{-1}) = 2 + 1 = 3 = \dim(E).$$

Plus précisément, la matrice A est semblable à la matrice diagonale

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}.$$

CHAPITRE 5

Oraux d'Informatique

Exercice 5.1 : Gestion de données démographiques

Une administration possède un fichier regroupant des informations sur les habitants d'une région. Ce fichier, appelé `Infos`, est constitué d'une liste de plusieurs milliers d'éléments, chaque élément étant une liste de 5 éléments de la forme :

[Numéro INSEE , Année de naissance , Genre , Nombre d'enfants , Profession]

Le numéro INSEE contient 13 chiffres, l'année de naissance en contient 4, le genre est un caractère `'M'` ou `'F'`, le nombre d'enfants est un entier naturel, la profession est une chaîne de caractères, par exemple `'Dentiste'` ou `'Agriculteur'`.
Toutes les fonctions demandées dans l'exercice prendront en entrée cette liste `Infos`.
1. Écrire une fonction Python donnant la proportion de femmes.
2. Créer une fonction qui renvoie la liste des numéros INSEE des hommes sans enfant.
3. Si nous sommes en 2024, calculer l'âge moyen de la population.
4. Écrire une fonction permettant de comparer la probabilité d'être vétérinaire selon que l'on soit un homme ou une femme.
5. Déterminer le nombre maximal d'enfants pour les personnes de cette liste, et retourner la liste des personnes ayant exactement ce nombre d'enfants.

Analyse du sujet et commentaires tactiques :
On manipule ici des **listes de listes**. Le parcours de la liste `Infos` peut s'effectuer de deux façons :
- D'une part on peut parcourir indice par indice, avec la commande

```
for i in range(n)
```

où n désigne la longueur de la liste `Infos`. L'indice i prendra la valeur 0, puis 1, ..., jusqu'à $n-1$. L'élément `L[i]` désigne alors le i-ème individu ; `L[i][0]` est le numéro INSEE de cet individu, `L[i][1]` son année de naissance, etc.
- D'autre part on peut parcourir élément par élément, avec la commande

```
for element in Infos .
```

Ici `element` désigne un élément de la liste, `element[0]` son numéro INSEE, `element[1]` son année de naissance, etc.

Dans cet exercice, puisqu'aucune question ne nécessite de repérer les indices de position des individus, on écrira les réponses avec cette deuxième méthode, mais la première aurait aussi pu être utilisée.

1.

On compte le nombre de femmes puis on divise par le nombre d'individus.

```
def femmes(Infos):
    C=0
    for element in Infos:
        if element[2]==2:
            C+=1
    return(C/len(Infos))
```

Explication détaillée du programme informatique :
On initialise un compteur C à 0, puis on parcourt la liste `Infos`.
Chaque fois qu'un individu est une femme, le compteur C augmente de 1.
Une fois sorti de la boucle, on retourne le quotient de C par le nombre total de personnes dans la liste.

Si l'étudiant préfère le parcours de la liste avec les indices, il faut écrire :

```
def femmes2(Infos):
    C=0
    n=len(Infos)
    for i in range(n):
        if Infos[i][2]==2:
            C+=1
    return(C/len(Infos))
```

2.

Voici un programme qui crée et retourne la liste des numéros INSEE des hommes sans enfant.

```
def pasdenfants(Infos):
    liste=[]
    for element in Infos:
        if element[2]==1 and element[3]==0:
            liste.append(element[0])
    return(liste)
```

Explication détaillée du programme informatique :
Au début on définit une liste vide.
Ensuite, avec la **boucle de répétition** `for`, on parcourt la liste `Infos` et,

si `element` est un homme et si son nombre d'enfants est 0, alors on ajoute `element[0]`, c'est-à-dire le numéro INSEE de `element`, à `liste`.
En fin de programme, après la boucle `for` on retourne `liste`.

3.

L'âge de `element` est `(2024-element[1])`. D'où le programme :

```
def agemoyen(Infos):
    S=0
    for element in Infos:
        S+=(2024-element[1])
    return(S/len(Infos))
```

Explication détaillée du programme informatique :
La somme S est initialisée à 0.
Puis on parcourt la liste et on ajoute à S l'âge de `element`.
Ensuite on retourne le quotient de S par le nombre total d'individus.

4. On va calculer le nombre d'hommes H, le nombre de femmes F, le nombre d'hommes vétérinaires $C1$ et le nombre de femmes vétérinaires $C2$. Il s'agit ensuite de retourner le quotient $\dfrac{C1}{H}$ qui est la probabilité d'être vétérinaire sachant que l'on est un homme, ainsi que le quotient $\dfrac{C2}{F}$ qui est la probabilité d'être vétérinaire sachant que l'on est une femme.

On aura besoin de quatre compteurs dans cette fonction.

```
def Veto(Infos):
    [C1,C2,H,F]=[0,0,0,0]
    for element in Infos:
        if element[2]==1:
            H+=1
            if element[4]=='veterinaire':
                C1+=1
        elif element[2]==2:
            F+=1
            if element[4]=='veterinaire':
                C2+=1
    return(C1/H,C2/F)
```

Explication détaillée du programme informatique :
Les quatre compteurs sont initialisés à 0.
Ensuite on parcourt la liste `Infos` avec la boucle `for`.
L'**instruction conditionnelle** `if` permet de mettre à jour les compteurs en examinant les différents cas :

- si `element` est un homme alors H augmente de 1, si de plus il est vétérinaire $C1$ augmente de 1 ;
- si `element` est une femme alors F augmente de 1, et si elle est vétérinaire alors $C2$ augmente de 1.

En sortie de boucle on retourne les quotients qui représentent les probabilités conditionnelles souhaitées.

5. Les algorithmes de **recherche de maximum** sont souvent demandés, les étudiants doivent les connaître.

On va créer une liste contenant les éléments réalisant le maximum.

```
def maxenfants(Infos):
    liste=[]
    M=0
    for element in Infos:
        if element[3]>M:
            M=element[3]
            liste=[element]
        elif element[3]==M:
            liste.append(element)
    return(M,liste)
```

Explication détaillée du programme informatique :
On initialise la valeur du maximum M à 0 et une liste vide.
On parcourt la liste `Infos`. Si le nombre d'enfants de `element` est strictement supérieur à la valeur actuelle du maximum M, alors on réactualise M et la liste ne contient que `element`. Sinon, si le nombre d'enfants de `element` est égal à la valeur actuelle du maximum M, alors on ajoute `element` à la liste.
Après la sortie de la boucle, on retourne M qui est la valeur du maximum, et `liste` qui contient les éléments réalisant ce maximum.