DS 3 Mathématiques

Les exercices et problèmes sont indépendants et peuvent être traités dans l'ordre souhaité. Dans l'exercice, les questions sont indépendantes entre elles. Dans les problèmes 1 à 3, les questions d'informatique peuvent être traitées indépendamment du problème.

Exercice

Calculer les sommes et produits suivants, en fonction de $n \in \mathbb{N}^*$.

1. $A_n = \displaystyle\sum_{k=0}^{2n} \left\lfloor \frac{k}{2} \right\rfloor^2$, en séparant les indices pairs et impairs.
2. $B_n = \displaystyle\sum_{0 \leqslant i \leqslant j \leqslant n} \binom{j}{i} 3^{i+j}$ avec le binôme de Newton.
3. $C_n = \displaystyle\sum_{k=0}^{n} \sin^2\left(\frac{k\pi}{2n}\right)$, avec le changement d'indice $j = n - k$ en montrant que $C_n = (n+1) - C_n$.
4. $D_n = \displaystyle\prod_{k=1}^{n} \exp\binom{n}{k}$, avec le binôme de Newton.

Problème 1

Le but est d'étudier la suite $(u_n)_{n\in\mathbb{N}}$ définie par

$$u_0 = 1 \quad \text{et} \quad \forall n \in \mathbb{N},\ u_{n+1} = \frac{u_n}{3 - u_n} \tag{1}$$

1. Écrire une fonction Python `suite_u_liste(n)` qui prend en argument un entier $n \in \mathbb{N}$ et qui renvoie la liste des n premiers termes de la suite.
2. On pose $f : x \mapsto \dfrac{x}{3-x}$.
   (a) Tracer le tableau de variations de la fonction f.
   (b) Donner le tableau de signe de $x \mapsto f(x) - x$.
   (c) Représenter graphiquement la fonction f, la droite d'équation $y = x$ et quelques-uns des premiers termes de la suite $(u_n)_{n\in\mathbb{N}}$. Que peut-on conjecturer sur les variations et la limite de la suite ?
   (d) Démontrer que $\forall n \in \mathbb{N},\ 0 < u_{n+1} < u_n \leqslant 1$ (en particulier, u_n est bien défini et ne s'annule pas).
3. Pour tout $n \in \mathbb{N}$, on pose $v_n = 1/u_n$.
   (a) Donner une relation de récurrence d'ordre 1 vérifiée par la suite $(v_n)_{n\in\mathbb{N}}$. Quel type de suite reconnait-on ?
   (b) Donner l'expression de v_n en fonction de $n \in \mathbb{N}$.
   (c) En déduire l'expression de u_n en fonction de $n \in \mathbb{N}$.
   (d) Déterminer la limite de u_n pour $n \to +\infty$.

Problème 2

Soit $n \in \mathbb{N}$, le but est de calculer la somme $S_n = \displaystyle\sum_{k=0}^{n} \binom{3n}{3k}$.

1. On commence par écrire un programme Python pour calculer cette somme :
   (a) Écrire une fonction `factoriel(n)` qui prend en argument en entier n supposé positif et qui renvoie la valeur de $n!$.
   (b) En déduire une fonction `binome(n, k)` qui prend en argument un entier n supposé positif et un nombre $k \in \mathbb{Z}$ et qui renvoie la valeur du coefficient binomial $\binom{n}{k}$.

(c) En déduire une fonction `somme(n)` qui prend en argument un entier n supposé positif et qui renvoie la valeur de S_n.

2. Calculer S_0, S_1 et S_2.
3. Soit j le nombre complexe $e^{2i\pi/3}$.
   (a) Montrer que $j^3 = 1$, que $j^2 = j^{-1} = \overline{j}$ et que $1 + j + j^2 = 0$.
   (b) Pour tout $p \in \mathbb{N}$, justifier qu'il existe un plus grand entier $k \in \mathbb{N}$ tel que $3k \leqslant p$, et qu'alors $p - 3k$ ne peut valoir que 0 ou 1 ou 2.
   (c) Montrer que
$$1 + j^p + j^{2p} = \begin{cases} 3 & \text{si } p = 3k \\ 0 & \text{si } p = 3k+1 \text{ ou } p = 3k+2 \end{cases} \tag{2}$$
   (d) Calculer les trois sommes suivantes, et donner le résultat sous forme algébrique :
$$A_n = \sum_{p=0}^{3n} \binom{3n}{p} \qquad B_n = \sum_{p=0}^{3n} j^p \binom{3n}{p} \qquad C_n = \sum_{p=0}^{3n} j^{2p} \binom{3n}{p} \tag{3}$$
   (e) Justifier que $A_n + B_n + C_n = 3S_n$, puis en déduire la valeur de S_n. Exprimer le résultat sans nombres complexes.

Problème 3

On note $\theta = \arccos(1/3)$ l'unique nombre réel dans $[0, \pi]$ tel que $\cos(\theta) = 1/3$. Le but est de démontrer que $\frac{\theta}{\pi}$ est un nombre irrationnel, c'est-à-dire ne peut pas s'écrire sous forme $\frac{p}{q}$ avec $p \in \mathbb{Z}$ et $q \in \mathbb{N}^*$. Ainsi θ lui-même ne peut pas s'écrire sous forme $\frac{p\pi}{q}$, et il n'y a pas d'angle remarquable dont le cosinus soit 1/3.
On raisonne donc par l'absurde en commençant par supposer qu'il existe $(p, q) \in \mathbb{Z} \times \mathbb{N}^*$ tels que $\theta = \frac{p\pi}{q}$.

1. (a) Calculer $\sin(\theta)$, puis donner la forme algébrique de $e^{i\theta}$.
   (b) En déduire qu'il existe un entier $N \in \mathbb{N}^*$ tel que $(1 + 2i\sqrt{2})^N = 3^N$.
2. Soient les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ définies par
$$a_0 = 1, \quad b_0 = 0, \quad \text{et} \quad \forall n \in \mathbb{N}, \begin{cases} a_{n+1} = a_n - 4b_n \\ b_{n+1} = 2a_n + b_n \end{cases} \tag{4}$$
   Montrer que $\forall n \in \mathbb{N}$, $(1 + 2i\sqrt{2})^n = a_n + i\sqrt{2}b_n$.
3. Écrire une fonction Python `suites_ab(n)`, prenant en argument en entier $n \in \mathbb{N}$, et qui renvoie la liste de deux éléments formée de a_n et de b_n.
4. Donner une relation de récurrence linéaire d'ordre 2 vérifiée par $(a_n)_{n\in\mathbb{N}}$, et de même pour $(b_n)_{n\in\mathbb{N}}$.
5. On rappelle qu'un entier est un *multiple de* 3 s'il s'écrit sous forme $3p$ avec $p \in \mathbb{Z}$; et que pour $m \in \mathbb{Z}$, si le nombre $2m$ est un multiple de 3 alors m lui-même est un multiple de 3. Montrer que pour tout $n \in \mathbb{N}$, a_n n'est pas un multiple de 3.
6. Conclure.

Problème d'informatique

Pour une liste `L`, la *liste miroir* de `L` est simplement la liste rangée en ordre inverse. Par exemple la liste miroir de `[6, 2, 1, 6, 7]` est `[7, 6, 1, 2, 6]`. Une liste sera dite *symétrique* si elle est égale à sa liste miroir. Par exemple les listes `[7, 2, 3, 2, 7]` ainsi que `[3, 8, 6, 6, 8, 3]` sont symétriques.

1. Pour une liste `L` de longueur n, quel est l'indice du dernier élément de `L` ? Et de l'avant dernier ? Quel est l'indice qui correspond au miroir de l'indice i ?
2. On souhaite écrire une fonction `est_symétrique(L)` qui prend en argument une liste `L` et qui renvoie `True` si `L` est symétrique, et `False` sinon. On considère la tentative suivante :

```python
def est_symétrique_raté(L):
    n = len(L)
    for i in range(n):
        if L[i] == L[n-1-i]:
            return True
        else:
            return False
```

   (a) Que renvoient `est_symétrique_raté([3, 7, 7, 3])` ? Et `est_symétrique_raté([2, 2, 8])` ? Et `est_symétrique_raté([3, 6, 7, 7, 1, 3])` ? Justifier.

   (b) Corriger la fonction (en la ré-écrivant sur sa copie, et en expliquant le problème) pour écrire la bonne fonction `est_symétrique(L)`.

3. Écrire une fonction `miroir(L)` qui renvoie une nouvelle liste `M` qui est la liste miroir de `L`.

Une liste `L` sera dite *pyramidale* si elle vérifie les trois conditions suivantes :

(i) `L` est de longueur impaire, disons de longueur $n = 2k + 1$ ($k \in \mathbb{N}$),

(ii) `L` est symétrique,

(iii) `L` est strictement croissante sur ses $k + 1$ premiers éléments — et donc automatiquement strictement décroissante sur ses $k + 1$ derniers éléments.

C'est le cas par exemple de la liste `[1, 2, 5, 6, 5, 2, 1]` (les termes sont croissants jusqu'au milieu puis décroissants, et forment donc une pyramide).
On rappelle que l'opération `n // 2` donne le quotient dans la division euclidienne de `n` par 2 et que `n % 2` donne le reste, qui est 0 si `n` est pair et 1 si `n` est impair.

4. Écrire une fonction `est_pyramidale(L)` qui renvoie `True` si la liste `L` est pyramidale et `False` sinon.

On suppose maintenant que la liste `L` est pyramidale. Une *marche* désigne l'écart entre deux éléments consécutifs de la liste — sur sa première moitié où elle est croissante. Par exemple pour `[1, 2, 5, 6, 5, 2, 1]` les marches sont simplement 1, 3 et 1 (car $2 - 1 = 1$, $5 - 2 = 3$ et $6 - 5 = 1$).

5. Écrire une fonction `marche1(L)` qui renvoie le nombre de marches exactement égales à 1.
6. Écrire une fonction `marche_max(L)` qui renvoie la plus grande marche de `L`. On pourra considérer que c'est 0 si la liste ne contient qu'un seul élément.