DS 3 Mathématiques
Correction

Exercice

1. On démarre comme suggéré :

$$A_n = \sum_{k=0}^{2n} \left\lfloor \frac{k}{2} \right\rfloor^2 \tag{1}$$

$$= \underbrace{\sum_{\substack{k=0 \\ \text{pair}}}^{2n} \left\lfloor \frac{k}{2} \right\rfloor^2}_{\text{poser } k=2j} + \underbrace{\sum_{\substack{k=0 \\ \text{impair}}}^{2n} \left\lfloor \frac{k}{2} \right\rfloor^2}_{\text{poser } k=2j+1} \tag{2}$$

$$= \sum_{j=0}^{n} \left\lfloor \frac{2j}{2} \right\rfloor^2 + \sum_{j=0}^{n-1} \left\lfloor \frac{2j+1}{2} \right\rfloor^2 \tag{3}$$

$$= \sum_{j=0}^{n} j^2 + \sum_{j=0}^{n-1} j^2 \tag{4}$$

En effet à droite le plus grand impair atteignable est $k = 2n-1$ (car $2n$ est pair) ce qui correspond à $j = n-1$. De plus $\frac{2j+1}{2} = j + \frac{1}{2}$, avec j entier, donc la partie entière est simplement égale à j.
Arrivé ici on peut remplacer (ou alors faire apparaitre deux fois $\sum_{j=1}^{n-1} j^2$ plus le morceau n^2), en remarquant que le terme d'indice 0 est nul :

$$A_n = \sum_{j=1}^{n} j^2 + \sum_{j=1}^{n-1} j^2 \tag{5}$$

$$= \frac{n(n+1)(2n+1)}{6} + \frac{(n-1)n(2n-1)}{6} \tag{6}$$

$$= \frac{n}{6}\Big((n+1)(2n+1) + (n-1)(2n-1)\Big) \tag{7}$$

$$= \frac{n}{6}\Big(4n^2 + 2\Big) \tag{8}$$

$$= \frac{n(2n^2+1)}{3} \tag{9}$$

2. Il faut sommer sur j d'abord puis sur i, car on voit une somme de $\binom{j}{i}$:

$$B_n = \sum_{0 \leqslant i \leqslant j \leqslant n} \binom{j}{i} 3^{i+j} \tag{10}$$

$$= \sum_{j=0}^{n} \left(\sum_{i=0}^{j} \binom{j}{i} 3^{i+j} \right) \tag{11}$$

$$= \sum_{j=0}^{n} \left(3^j \sum_{i=0}^{j} \binom{j}{i} 3^i \right) \tag{12}$$

Ici c'est presque fini car sous la somme on reconnait le binôme de Newton pour $(1+3)^j$.

$$B_n = \sum_{j=0}^{n} \left(3^j \times (1+3)^j\right) \tag{13}$$
$$= \sum_{j=0}^{n} 3^j 4^j \tag{14}$$
$$= \sum_{j=0}^{n} 12^j \tag{15}$$
$$= \frac{1-12^n}{1-12} \qquad \text{(suite géométrique de raison 12)} \tag{16}$$
$$= \frac{12^n - 1}{11} \tag{17}$$
$$\tag{18}$$

3. Comme suggéré, on effectue le changement d'indice $j = n - k$, alors $k = 0$ correspond à $j = n$ et $k = n$ correspond à $j = 0$:

$$C_n = \sum_{k=0}^{n} \sin^2\left(\frac{k\pi}{2n}\right) \tag{19}$$
$$= \sum_{j=0}^{n} \sin^2\left(\frac{(n-j)\pi}{2n}\right) \tag{20}$$
$$= \sum_{j=0}^{n} \sin^2\left(\frac{\pi}{2} - \frac{j\pi}{2n}\right) \tag{21}$$
$$= \sum_{j=0}^{n} \cos^2\left(\frac{j\pi}{2n}\right) \tag{22}$$

par la formule de trigonométrie $\sin(\frac{\pi}{2} - \theta) = \cos(\theta)$. Mais alors on a aussi $\cos^2(\theta) = 1 - \sin^2(\theta)$, d'où

$$C_n = \sum_{j=0}^{n} \left(1 - \sin^2\left(\frac{j\pi}{2n}\right)\right) \tag{23}$$
$$= \sum_{j=0}^{n} 1 - \sum_{j=0}^{n} \sin^2\left(\frac{j\pi}{2n}\right) \tag{24}$$
$$= (n+1) - C_n \tag{25}$$

car à droite c'est la même somme qui apparait, quitte à renommer l'indice j en k.
(Alternativement, cela revient à dire qu'on calcule $\sum_{j=0}^{n}\left(\sin^2\left(\frac{j\pi}{2n}\right) + \sum_{j=0}^{n}\sin^2\left(\frac{j\pi}{2n}\right)\right)$ de deux façons différentes, un peu comme dans la méthode de Gauss pour les suites arithmétiques.)
On en conclue donc que $2C_n = n+1$ donc $C_n = \frac{n+1}{2}$.

4. On transforme un produit d'exponentielles en exponentielle d'une somme, et ensuite il suffit de faire apparaitre le terme d'indice $k = 0$ pour avoir la bonne formule du binôme :

$$D_n = \prod_{k=1}^{n} \exp\binom{n}{k} \tag{26}$$
$$= \exp\left(\sum_{k=1}^{n} \binom{n}{k}\right) \tag{27}$$
$$= \exp\left(\sum_{k=0}^{n} \binom{n}{k} - 1\right) \qquad \text{(terme pour } k = 0\text{)} \tag{28}$$
$$= \exp\left((1+1)^n - 1\right) \tag{29}$$
$$= \exp(2^n - 1) \tag{30}$$

Problème 1

1. Standard, devrait être sans commentaire. On veut les n premiers termes, c'est à dire les termes de u_0 à u_{n-1}, et c'est bien `range(n)` qui parcourt tous ces indices (mais u_0 est déjà donné).

```
def suite_u_liste(n):
    L = [0] * n
    L[0] = 1
    for i in range(1, n):
        L[i] = L[i-1] / (3 - L[i-1])
    return L
```

2. (a) f n'est pas définie si $x = 3$. Sur $\mathbb{R} \setminus \{3\}$, f est dérivable et

$$\forall x \in \mathbb{R} \setminus \{3\}, \quad f'(x) = \frac{1 \times (3-x) - x \times (-1)}{(3-x)^2} \tag{31}$$
$$= \frac{3 - x + x}{(3-x)^2} \tag{32}$$
$$= \frac{3}{(3-x)^2} > 0 \tag{33}$$

mais attention, f n'est pas strictement croissante sur tout $\mathbb{R} \setminus \{3\}$ car il y a une valeur interdite ; f est strictement croissante sur $]-\infty, 3[$ et sur $]3, +\infty[$.

x	$-\infty$		3		$+\infty$
$f'(x)$	$+$		‖	$+$	
$f(x)$	-1	↗	‖ $+\infty$	$-\infty$ ↗	-1

(34)

Placer les limites n'est pas non plus très difficile : en $\pm\infty$ on écrit $\frac{x}{3-x} = \frac{x}{x(\frac{3}{x}-1)} = \frac{1}{\frac{3}{x}-1}$ et la limite en bas est -1 ; et en 3 il faut distinguer les limites en 3^- (juste avant 3) où $3-x > 0$ donc $\lim_{x\to 3^-} \frac{1}{3-x} = +\infty$, et de même en 3^+ on a $3 - x < 0$ et donc $\lim_{x\to 3^+} = -\infty$.

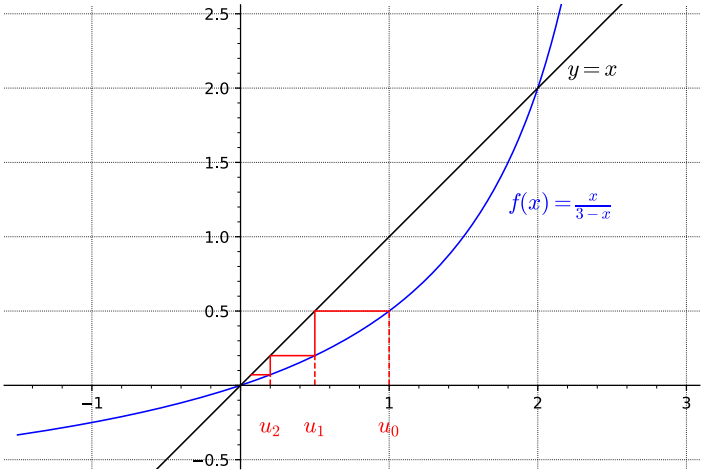
(b) On calcule simplement

$$f(x) - x = \frac{x}{3-x} - x \tag{35}$$
$$= \frac{x - x(3-x)}{3-x} \tag{36}$$
$$= \frac{x(x-2)}{3-x} \tag{37}$$

Le tableau de signe apparait alors :

x	$-\infty$		0		2		3		$+\infty$
x		$-$	0	$+$		$+$		$+$	
$x-2$		$-$		$-$	0	$+$		$+$	
$3-x$		$+$		$+$		$+$	0	$-$	
$f(x)$		$+$	0	$-$	0	$+$	‖	$-$	

(38)



(c) On conjecture que la suite $(u_n)_{n\in\mathbb{N}}$ reste dans l'intervalle $[0,1]$, est strictement décroissante et converge vers 0.

(d) On démontre d'un seul coup tout ce dont on a besoin ; à ce stade presque tout est déjà donné par les tableaux de signe et de variations tracés. Soit pour $n \in \mathbb{N}$ l'hypothèse $\mathcal{P}(n)$: « $0 < u_{n+1} < u_n \leqslant 1$ ».
$n = 0$: on a $u_0 = 1$ donc $0 < u_0 \leqslant 1$. Il reste à calculer $u_1 = f(u_0)$ qui est donc égal à $\frac{1}{3-1} = \frac{1}{2}$ donc on a bien $0 < u_1 < u_0$.
On pouvait aussi se passer de ce calcul : sachant $0 < u_0 < 2$, avec f strictement croissante sur cet intervalle et $f(0) = 0$ et $f(2) = 2$ (car ce sont des solutions de $f(x) = x$), on trouve alors $0 < f(u_0) < 2$. Et sachant que encore une fois pour $0 < u_0 < 2$ on a $f(x) - x < 0$ on en déduit $f(u_0) < u_0$. Mais $u_1 = f(u_0)$.
Hérédité : soit $n \in \mathbb{N}$, supposons $\mathcal{P}(n)$. Alors là encore d'après tout ce qui a été dit avant, sachant $0 < u_{n+1} < u_n \leqslant 1$ nous sommes sur un intervalle où f est strictement croissante donc on obtient $f(0) < f(u_{n+1}) < f(u_n) \leqslant f(1)$. Mais $f(0) = 0$ et $f(1) < 1$. On obtient donc bien $0 < u_{n+2} < u_{n+1} \leqslant 1$ et ceci démontre $\mathcal{P}(n+1)$.

3. (a) On a alors $u_n = \frac{1}{v_n}$, on remplace simplement dans la relation de récurrence :

$$\frac{1}{v_{n+1}} = \frac{\frac{1}{v_n}}{3 - \frac{1}{v_n}} \tag{39}$$

$$= \frac{1}{3v_n - 1} \tag{40}$$

d'où on déduit $v_{n+1} = 3v_n - 1$. Il s'agit d'une relation arithmético-géométrique.

(b) Le point fixe $c \in \mathbb{R}$ vérifie $c = 3c - 1$, d'où $2c = 1$ d'où $c = \frac{1}{2}$. De

$$v_{n+1} = 3v_n - 1 \tag{41}$$

$$c = 3c - 1 \tag{42}$$

on déduit $v_{n+1} - c = 3(v_n - c)$: la suite des $(v_n - c)_{n\in\mathbb{N}}$ est géométrique de raison 3 et ainsi $\forall n \in \mathbb{N}$, $v_n - c = 3^n(v_0 - c)^n$. Ici rappelons que $u_0 = 1$ donc $v_0 = 1$, et $v_0 - c = \frac{1}{2}$. On en déduit $v_n = 3^n \times \frac{1}{2} + \frac{1}{2}$.

(c) On en déduit alors

$$\forall n \in \mathbb{N}, \quad u_n = \frac{1}{3^n \times \frac{1}{2} + \frac{1}{2}} \tag{43}$$

$$= \frac{2}{3^n + 1} \tag{44}$$

(d) Quand $n \to +\infty$ alors $3^n \to +\infty$ et $3^n + 1 \to +\infty$ aussi. Par inverse on en déduit donc bien $\lim_{n\to+\infty} = 0$.

Problème 2

1. (a) Classique, à connaitre sans hésiter !

```
def factoriel(n):
    # u représente (i-1)! en début de boucle
    u = 1
    for i in range(1, n+1):
        u = n * u
    return u
```

Si $n = 0$ ainsi que $n = 1$ cela renvoie bien 1, et lors de la dernière itération on multiplie par n, les indices et bornes sont bons.

(b) Tout simplement avec la formule $\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$. Mais attention, telle que la question est posée, il faut aussi considérer le cas où on n'a pas $0 \leqslant k \leqslant n$, et où $\binom{n}{k} = 0$.

```
def binome(n, k):
    if k < 0 or k > n:
        return 0
    return factoriel(n) / (factoriel(k) * factoriel(n-k))
```

Ce n'est clairement pas la façon la plus efficace ; de plus il est préférable d'utiliser la division euclidienne ici `//` sinon les valeurs renvoyées seront de type nombre à virgule flottante. Passons sur ces détails pour l'instant.

(c) Sans commentaire, c'est une somme !

```
def somme(n):
    # variable accumulatrice pour la somme
    S = 0
    for k in range(0, n+1):
        S = S + binome(3*n, 3*k)
    return S
```

2. On a facilement $S_0 = \binom{0}{0} = 1$ et $S_1 = \binom{3}{0} + \binom{3}{3} = 1 + 1 = 2$. Enfin $S_2 = \binom{6}{0} + \binom{6}{3} + \binom{6}{6}$. Mais on a alors besoin d'un petit triangle de Pascal au brouillon pour trouver $\binom{6}{3} = 20$. Et donc $S_2 = 1 + 20 + 1 = 22$.

3. (a) On calcule $j^3 = \left(e^{2i\pi/3}\right)^3 = e^{2i\pi} = 1$. On a aussi $j^{-1} = \frac{1}{e^{2i\pi/3}} = e^{-2i\pi/3} = \overline{j}$. Mais remarquons que $-\frac{2i\pi}{3} + 2i\pi = \frac{4i\pi}{3}$ et donc $e^{-2i\pi/3} = e^{4i\pi/3}$, ce dernier étant j^2.
Enfin l'égalité $1 + j + j^2 = 0$ peut se voir de nombreuses façons (y compris en passant par la forme algébrique). C'est aussi un cas particulier de sommes des racines de l'unité, la suite est géométrique de raison j et donc $1 + j + j^2 = \frac{1-j^3}{1-j} = 0$.

(b) Soit $p \in \mathbb{N}$. L'ensemble des entiers $k \in \mathbb{N}$ tels que $3k \leqslant p$ est une partie de $\mathbb{N}$, non-vide (contenant 0) et majorée (car $k \leqslant \frac{p}{3}$), elle admet donc un maximum.
Pour ce k là, posons $r = p - 3k$, c'est un entier et $r \geqslant 0$. Montrons par l'absurde que $r \leqslant 2$. Si on avait $r \geqslant 3$, alors on aurait que $3(k+1) = 3k + 3 \leqslant p$ car $p - 3k = r \geqslant 3$, ce qui est en contradiction avec le fait que k est le plus grand possible ! On en déduit donc que $r < 3$, et comme r est entier, c'est 0 ou 1 ou 2.
Tout ceci montre que tout nombre entier positif est de la forme $3k$ ou $3k+1$ ou $3k+2$ (c'est l'analogue des nombres pairs et impairs, mais pour la division par 3).

(c) On écrit alors dans chacun des cas : si $p = 3k$ alors

$$1 + j^p + j^{2p} = 1 + j^{3k} + j^{6k} = 1 + (j^3)^k + (j^3)^{2k} = 1 + 1 + 1 = 3 \tag{45}$$

mais si $p = 3k + 1$ alors

$$1 + j^p + j^{2p} = 1 + j^{3k+1} + j^{6k+2} = 1 + (j^3)^k \times j + (j^3)^{2k} \times j^2 = 1 + j + j^2 = 0 \tag{46}$$

et si $p = 3k + 2$ alors

$$1 + j^p + j^{2p} = 1 + j^{3k+2} + j^{6k+4} = 1 + (j^3)^k \times j^2 + (j^3)^{2(k+1)} \times j = 1 + j^2 + j = 0 \tag{47}$$

(d) On reconnait pour chacune d'entre elles le binôme de Newton (avec $j^{2p} = (j^2)^p$ pour la dernière)

$$A_n = (1+1)^{3n} = 2^{3n} \qquad B_n = (1+j)^{3n} \qquad C_n = (1+j^2)^{3n} \tag{48}$$

Mais sous cette forme ce n'est pas satisfaisant ! Pour simplifier B_n la méthode standard est d'utiliser une factorisation par angle moitié, sous la puissance.

$$\begin{aligned} B_n &= \left(1 + e^{2i\pi/3}\right)^{3n} && (49)\\ &= \left(e^{i\pi/3}\left(e^{-i\pi/3} + e^{i\pi/3}\right)\right)^{3n} && (50)\\ &= \left(e^{i\pi/3} \times 2\cos(\pi/3)\right)^{3n} && (51)\\ &= e^{in\pi} \times 2^{3n}\cos^{3n}(\pi/3) && (52)\\ &= (\cos(n\pi) + i\sin(n\pi)) \times 2^{3n}\cos^{3n}(\pi/3) && (53)\\ &= 2^{3n}\cos(n\pi)\cos^{3n}(\pi/3) + 2^{3n} i\sin(n\pi)\cos^{3n}(\pi/3) && (54) \end{aligned}$$

On peut simplifier encore sachant que $\cos(\pi/3) = \frac{1}{2}$, $\sin(n\pi) = 0$ et $\cos(n\pi) = (-1)^n$, on a alors tout simplement

$$B_n = 2^{3n} \times (-1)^n \times \left(\frac{1}{2}\right)^{3n} + 0 = (-1)^n \tag{55}$$

On pouvait aussi remarquer plus tôt que $2\cos(\pi/3) = 1$ et que $e^{in\pi} = (-1)^n$; on bien mettre dès le départ $1+j$ sous forme exponentielle en passant d'abord par sa forme algébrique et trouver $1+j = e^{i\pi/3}$.
Pour C_n on peut reprendre le même type de calcul, ou bien remarque que comme $j^2 = \overline{j}$ on a automatiquement $C_n = \overline{B_n}$. On trouve donc de toute façon

$$C_n = 2^{3n}\cos(n\pi)\cos^{3n}(\pi/3) - 2^{3n} i\sin(n\pi)\cos^{3n}(\pi/3) = (-1)^n \tag{56}$$

(e) Si on calcule, on trouve

$$\begin{aligned} A_n + B_n + C_n &= \sum_{p=0}^{3n}\binom{3n}{p} + \sum_{p=0}^{3n} j^p\binom{3n}{p} + \sum_{p=0}^{3n} j^{2p}\binom{3n}{p} && (57)\\ &= \sum_{p=0}^{3n}\left(\binom{3n}{p} + j^p\binom{3n}{p} + j^{2p}\binom{3n}{p}\right) && (58)\\ &= \sum_{p=0}^{3n}\left(1 + j^p + j^{2p}\right)\binom{3n}{p} && (59) \end{aligned}$$

Sous la somme, on a trois cas possibles sur p selon $p = 3k$ ou $3k+1$ ou $3k+2$, et on a vu que dans ces deux derniers cas le termes sous la somme est nul à cause de $1 + j^p + j^{2p} = 0$. Il ne reste donc que les termes pour $p = 3k$. Mais (exactement comme dans la séparation des indices pairs et impairs), cela signifie qu'on peut en fait sommer sur k, le plus petit k possible étant 0 et le plus grand étant n pour que $3k = 3n$. Dans ce cas $1 + j^p + j^{2p} = 3$. On trouve donc

$$\begin{aligned} A_n + B_n + C_n &= \sum_{k=0}^{n} 3 \times \binom{3n}{3k} && (60)\\ &= 3 \times \sum_{k=0}^{n}\binom{3n}{3k} && (61)\\ &= 3S_n && (62) \end{aligned}$$

On en déduit donc $S_n = \frac{1}{3}(A_n + B_n + C_n)$.
À cette question ou à la précédente, il faut bien avoir simplifié B_n. Il n'était pas si évident que B_n et C_n sont réels, mais comme $C_n = \overline{B_n}$ on doit avoir les parties imaginaires qui se compensent et donc $B_n + C_n$ est réel. On trouve, en simplifiant à cette question ou lors de la précédente :

$$S_n = \frac{1}{3}\left(2^{3n} + (-1)^n + (-1)^n\right) = \frac{2^{3n} + 2 \times (-1)^n}{3} \tag{63}$$

C'est d'ailleurs cohérent avec nos calculs notamment $S_1 = \frac{2^3-2}{3} = 2$ et $S_2 = \frac{2^6+2}{3} = 22$ car $2^6 = 64$.

Problème 3

1. (a) On sait que $\cos(\theta) = \frac{1}{3}$. Par la relation de Pythagore $\sin^2(\theta) = 1 - \cos^2(\theta) = 1 - \frac{1}{9} = \frac{8}{9}$. On en déduit donc
$$\sin(\theta) = \pm\sqrt{\frac{8}{9}} = \pm\frac{\sqrt{8}}{3} = \pm\frac{2\sqrt{2}}{3} \tag{64}$$
Mais par l'énoncé $\theta \in [0, \pi]$ donc $\sin(\theta) \geqslant 0$ donc il faut choisir la racine avec un signe + : $\sin(\theta) = \frac{2\sqrt{2}}{3}$.
On en déduit donc $e^{i\theta} = \cos(\theta) + i\sin(\theta) = \frac{1}{3} + i\frac{2\sqrt{2}}{3} = \frac{1+2i\sqrt{2}}{3}$.

   (b) Supposant que $\theta = \frac{p\pi}{q}$, on a $e^{i\pi p/q} = \frac{1+2i\sqrt{2}}{3}$ donc $3e^{i\pi p/q} = 1+2i\sqrt{2}$. Pour tout entier $n \in \mathbb{N}$, si on élève à la puissance n des deux côtés il vient l'égalité $3^n e^{ni\pi p/q} = (1+2i\sqrt{2})^n$. L'idée est alors de prendre n tel que $e^{ni\pi p/q} = 1$, et c'est bien le cas si np/q est un multiple de 2 (car alors si $np/q = 2m$ on a e^{2impi} qui fait 1). On peut donc simplement prendre $N = 2q$ (ou tout multiple de celui-ci).

2. On démontre ce résultat aisément par récurrence. On pose $\mathcal{P}(n)$: « $(1+2i\sqrt{2})^n = a_n + i\sqrt{2}b_n$ ».
Pour $n = 0$ on a $(1+2i\sqrt{2})^n = 1 = 1 + 0i$ ce qui est cohérent avec $a_0 = 1$ et $b_0 = 0$. Cela démontre $\mathcal{P}(0)$.
Soit $n \in \mathbb{N}$, supposons $\mathcal{P}(n)$. Alors
$$\begin{aligned}
(1+2i\sqrt{2})^{n+1} &= (1+2i\sqrt{2}) \times (1+2i\sqrt{2})^n && (65)\\
&= (1+2i\sqrt{2}) \times (a_n + i\sqrt{2}b_n) && (\text{par } \mathcal{P}(n)) \quad (66)\\
&= (a_n - 2 \times (\sqrt{2})^2 b_n) + i(2\sqrt{2}a_n + \sqrt{2}b_n) && (67)\\
&= (a_n - 4b_n) + i\sqrt{2}(2a_n + b_n) && (68)\\
&= a_{n+1} + i\sqrt{2}b_{n+1} && (69)
\end{aligned}$$
Ceci termine de démontrer $\mathcal{P}(n+1)$.

   Remarque 1. La méthode présentée ici aurait aussi permis de démontrer l'existence des suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ en découvrant leur relation de récurrence, sans forcément qu'elle soit donnée à l'avance. On aurait alors rédigé « montrons par récurrence sur $n \in \mathbb{N}$ qu'il existe des nombres a_n, b_n tels que… » et lors de la dernière ligne, on écrirait qu'on doit poser $a_{n+1} = a_n - 4b_n$ et $b_{n+1} = 2a_n + b_n$. La conclusion de la récurrence est que cela construit des suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$.
   Avec la formule du binôme de Newton, on pourrait aussi développer $(1+2i\sqrt{2})^n$ et séparer les parties réelles et imaginaires, ce qui reviendrait à séparer les termes d'indices pairs et impairs. Cela donne d'autres formules pour les suites, mais qui n'aident pas beaucoup.

3. Il s'agit simplement de fusionner les programmes standards de calcul de suites, avec une variable `a` qui représente a_i dans la boucle et `b` qui représente b_i, et de renvoyer la liste de deux éléments `[a, b]`. Le programme ressemble à :

```
def suites_ab(n):
    a = 1
    b = 0
    for i in range(n):
        # attention, ceci ne marche pas
        a = a - 4*b
        b = 2*a + b
    return [a, b]
```

   Tel quel cela ne marche pas car à la ligne `a = a - 4*b` l'ancienne valeur de `a` est perdue et ne peut donc pas être utilisée à la ligne suivante, c'est exactement comme pour le problème de l'échange de deux variables ou pour le calcul de suites récurrences d'ordre 2. Il faut donc passer par une variable intermédiaire, par exemple :

```
def suites_ab(n):
    a = 1
    b = 0
    for i in range(n):
        # nouvelle valeur de a
        t = a - 4*b
        b = 2*a + b
        a = t
    return [a, b]
```

4. On écrit d'abord $a_{n+2} = a_{n+1} - 4b_{n+1}$. On remplace alors les a_{n+1} et b_{n+1} donc

$$a_{n+2} = (a_n - 4b_n) - 4(2a_n + b_n) \tag{70}$$
$$= -7a_n - 8b_n \tag{71}$$

puis on remplace à nouveau avec la première ligne $b_n = \frac{a_n - a_{n+1}}{4}$ soit $8b_n = 2(a_n - a_{n+1})$ donc

$$a_{n+2} = -7a_n - 2(a_n - a_{n+1}) \tag{72}$$
$$= -9a_n + 2a_{n+1} \tag{73}$$

autrement dit $a_{n+2} - 2a_{n+1} + 9a_n = 0$.

On fait de même pour b_n et on trouve la même relation de récurrence : $b_{n+2} = 2a_{n+1} + b_{n+1}$ donc $b_{n+1} = 2(a_n - 4b_n) + (2a_n + b_n) = 4a_n - 7b_n$ où on remplace ensuite a_n par $\frac{b_{n+1} - b_n}{2}$ donne $b_{n+2} = 2b_{n+1} - 9b_n$.

5. Si on tente d'exprimer les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ en fonction de n, on retombe sur le nombre $e^{i\theta}$ comme solution de l'équation caractéristique et sur $(1 + 2i\sqrt{2})^n$: cela ne nous avance pas, il faut regarder avec la relation de récurrence. Dans celle-ci on remarque que le terme $9a_n$ est toujours un multiple de 3, il semble donc que si a_{n+1} était aussi multiple de 3 alors a_{n+2} le serait, et tous les termes de la suite, ce qui semble problématique sur les premiers termes. On voit aussi, avec le principe rappelé dans l'énoncé, que si a_{n+2} était multiple de 3 alors $2a_{n+1}$ devrait l'être et donc a_{n+1} aussi !

Mais pour formaliser cela le plus clair est de commencer par rédiger une récurrence double. On pose donc $\mathcal{P}n$: « a_n n'est pas un multiple de 3 ».

$n = 0$: $a_0 = 1$ n'est pas un multiple de 3.

$n = 1$: on calcule à la main $a_1 = 1$, qui n'est toujours pas un multiple de 3.

Soit $n \in \mathbb{N}$, supposons $\mathcal{P}(n)$ et $\mathcal{P}(n+1)$. Montrons par l'absurde que a_{n+2} n'est pas un multiple de 3. Supposons que a_{n+2} le soit, on écrirait alors

$$2a_{n+1} = a_{n+2} + 9a_n \tag{74}$$

Ceci est multiple de 3, car a_{n+2} l'est et un multiple de 9 l'est toujours. On en déduit donc (principe rappelé dans l'énoncé) que a_{n+1} doit être un multiple de 3. Ce qui est en contradiction avec $\mathcal{P}(n+1)$.

Remarque 2. Une autre rédaction possible serait de commencer par l'absurde en supposant qu'il existe un entier n tel que a_n n'est pas multiple de 3 et de considérer le plus petit tel n. Alors avec a_0 et a_1, n est nécessairement plus grand que 2. La relation de récurrence écrite avec a_n en fonction de a_{n-1} et a_{n-2} donne $2a_{n-1} = a_n + 9a_{n-2}$ d'où on conclurait comme précédemment que a_{n-1} est multiple de 3, mais il serait plus petit que le plus petit entier, ce qui est absurde.

6. D'après la question 1.(b), on a montré qu'en supposant $\frac{\theta}{\pi}$ rationnel on trouvait $N \in \mathbb{N}^*$ tel que $(1+2i\sqrt{2})^N = 3^N$ c'est à dire $a_N + i\sqrt{2}b_N = 3^N$ soit $a_N = 3^N$ et $b_N = 0$. Mais pour cet entier N on a bien sûr que a_N est un multiple de 3. Ceci est une contradiction. C'est que l'hypothèse sur $\frac{\theta}{\pi}$ était fausse, et donc $\frac{\theta}{\pi}$ est irrationnel.

Problème d'informatique

1. Si `L` est de longueur n, le dernier élément a pour indice $n-1$. L'avant dernier est donc à l'indice $n-2$. En général l'élément miroir de celui à l'indice i est à $n-1-i$.

2. `est_symétrique_raté([3, 7, 7, 3])` renvoie bien `True` : la liste est de longueur 4, on teste l'égalité entre les éléments d'indices 0 et 3 (donc les deux `3`) qui est bien vraie et cela arrête la boucle et renvoie `True`.

   `est_symétrique_raté([2, 2, 8])` renvoie `False` : la liste est de longueur 3 et on teste d'abord l'égalité entre les éléments d'indice 0 et 2 c'est-à-dire les éléments `2` et `8`. Cela renvoie `False`.

   `est_symétrique_raté([3, 6, 7, 7, 1, 3])` : la liste est de longueur 6, on teste d'abord l'égalité entre les éléments d'indice 0 et 5 c'est-à-dire les deux `3`. La boucle s'arrête donc et la fonction renvoie `True`.

3. Le problème est que `return` arrête la boucle. Or, si le premier et le dernier élément sont égaux, cela ne signifie pas que la liste est symétrique, mais qu'il faut continuer à tester. Mais on ne peut pas vraiment dire de continuer à tester : ce qu'on fait c'est qu'on arrête la fonction dès qu'on trouve en contre-exemple montrant que la liste n'est pas symétrique, et sinon on arrive en fin de boucle sans avoir arrêté la fonction plus tôt donc c'est que la liste était bien symétrique.

```
def est_symétrique(L):
    n = len(L)
    for i in range(n):
        # si cela se produit : nous sommes certains que la liste
        # *n'est pas* symétrique
        if L[i] != L[n-1-i]:
            return False
    # si nous arrivons ici c'est que nous n'avons pas quitté la boucle
    # donc que oui, la liste est symétrique
    return True
```

   C'est le même principe pour tester si tous les éléments d'une liste vérifient une certaine propriété, ou pour tester si une liste est croissante.

4. Connaissant l'indice du miroir, écrire la fonction n'est pas très difficile. Il faut bien initialiser une liste de zéros au départ.

```
def miroir(L):
    n = len(L)
    M = [0] * n
    for i in range(n):
        M[i] = L[n-1-i]
    return M
```

   Un programmeur expérimenté écrirait plutôt cela en compréhension :

```
M = [L[n-1-i] for i in range(n)]
```

   C'est la méthode à préférer quand on a directement une formule pour `M[i]` en fonction de `i`. On l'obtient aussi avec la syntaxe `M = L[::-1]` mais bien sûr dans une telle question ce n'est pas ce qui est attendu...

   Remarquez aussi que les deux premières questions du problème permettent de se remettre en tête la syntaxe des listes et leur convention de numérotation. Les suivantes demandent d'écrire des fonctions sans aide, mais on peut s'inspirer du style proposé là !

5. Là encore il faut tester les conditions successivement et renvoyer `False` quand elles ne sont pas vérifiées. Le nombre k tel que $n = 2k+1$ se retrouve avec `n // 2`. Ensuite il faut tester la strict croissante, c'est-à-dire la condition `L[i] < L[i+1]` (dont le contraire est `L[i] >= L[i+1]`) sur la liste des $k+1$ premiers termes (du début jusqu'à l'élément au milieu qui est sommet de la pyramide, inclus, qui est d'indice k). C'est `for i in range(k+1)` qui parcourt ces indices mais comme on considère `L[i+1]` il faut aller un cran avant. Vérifier que pour une liste de longueur 3 (donc $k = 1$) il y a une comparaison à faire, et pour une liste de longueur 1 (donc $k = 0$) il n'y en a pas : c'est bien `range(k)`.

```
def est_pyramidale(L):
    n = len(L)
    if n % 2 == 0:
        # longueur paire
        return False
    k = n // 2
    if not est_symétrique(L):
        # n'est pas symétrique
        return False
    for i in range(k):
        if L[i] >= L[i+1]:
            # n'est pas strictement croissante
            return False
    # si on arrive enfin là, c'est bon
    return True
```

6. Il s'agit d'une fonction pour compter, avec une boucle comme précédemment pour parcourir les `L[i+1] - L[i]`.

```
def marche1(L):
    n = len(L)
    k = n // 2
    c = 0
    for i in range(k):
        if L[i+1] - L[i] == 1:
            c = c + 1
    return c
```

Remarque 3. On suppose que la liste est déjà pyramidale, et ce n'est jamais à vous de le vérifier et de le tester dans la fonction. Tout ce qu'on demande, c'est qu'elle donne le bon résultat si la liste est effectivement pyramidale. C'est une façon de penser assez différente de l'informatique théorique à l'écrit par rapport à l'informatique concrète en salle de TP, où on voudrait avoir une belle interface qui dise à l'utilisateur « attention, ta liste n'est pas correcte » mais ceci prend beaucoup de temps et n'a aucun rapport avec l'algorithmique !

7. Idem, il s'agit d'une simple fonction pour calculer un maximum, mais sur les `L[i+1] - L[i]`.

```
def marche_max(L):
    n = len(L)
    k = n // 2
    M = 0
    for i in range(k):
        if L[i+1] - L[i] > M:
            M = L[i+1] - L[i]
    return M
```