LYCEE CHARLEMAGNE
Friday the $13^{th}$
*M.P.S.I.2*

2024 2025

# DS04

Il y a 720 anagrammes du nombre 123456. Votre mission est de calculer leur somme. 5 pt.
Première méthode : en moyenne, combien valent ces 720 nombres (deux à deux ?).
Deuxième méthode : un programme Python pour les reconnaître et les sommer ?
Troisième méthode : chaque nombre est un $\sum_{k=1}^{6} \sigma(k).10^{k-1}$ pour $\sigma$ dans $S_6$.
Vous avez réussi la première épreuve ? Alors, la somme de leurs carrés. 4 pt.

## Convolution de suites.

I∼0) $(E, +, \times, .)$ *est l'algèbre des suites réelles*[1]. *On définit un opérateur de translation* $T$, *un opérateur de dérivation* $\Delta$ *et une loi de convolution* $*$.
*Pour toute suite* $a$, *la suite* $T(a)$ *est la suite définie par* $\forall n,\ (T(a))_n = a_{n+1}$ *(bref :* $T = (a_0,\ a_1,\ a_2, \ldots) \longmapsto (a_1,\ a_2,\ a_3, \ldots)$*).* Montrez que la translatée d'une suite géométrique est encore géométrique. 1 pt.
I∼1) *Pour toute suite* $a$, *sa « dérivée »* $\Delta(a)$ *est la suite définie par* $\forall n,\ (\Delta(a))_n = a_{n+1} - a_n$ *(la notation* $\Delta(a_n)$ *n'a pas plus de sens que* $(f(x))'$, *on est d'accord).* Montrez que la dérivée d'une suite géométrique est encore géométrique. 1 pt.
I∼2) Dérivez pour $k$ fixé la suite $(\binom{n}{k})$. 2 pt.
I∼3) Montrez : $\forall a,\ \forall n,\ (\Delta^2(a))_n = a_{n+2} - 2.a_{n+1} + a_n, \forall a,\ \forall n,\ (\Delta^3(a))_n = a_{n+3} - 3.a_{n+2} + 3.a_{n+1} - a_n$. 2 pt.
I∼4) Proposez une formule générale pour $\Delta^k(a)$ (sans la démontrer). 1 pt.
I∼5) Un élève propose une preuve de ce que vous avez avancé en écrivant $\Delta^k = (T - Id)^k = \sum_{i=0}^{k} \binom{k}{i} \ldots$ ; qu'en pensez vous ? 2 pt.

II∼0) *Pour tout couple de suites* $(a,\ b)$, *la suite* $a * b$ *définie par* $(a * b)_n = \sum_{k=0}^{n} a_{n-k}.b_k$ *(d'accord, là aussi,* $(a_n) * (b_n)$ *n'a aucun sens).* Montrez que la loi $*$ est commutative et associative. 3 pt.
II∼1) Montrez que la suite $(1, 0, 0, \ldots)$ (suite géométrique de raison 0 si on réfléchit bien) est élément neutre de $*$, qu'on notera $e$. 1 pt.
II∼2) Écrivez une procédure Python qui prend en entrée deux listes a et b (suites dont on n'a qu'un nombre fini de termes), supposées de même longueur N et retourne leur convolée a*b (en tout cas les N premiers termes de la suite a*b). 2 pt.
exemple `convole([0, 1, 2, 3, 4], [1, 2, 4, 8, 16]) -> [0, 1, 4, 11, 26]`
II∼3) $a$ est pour cette question la suite constante égale à 1. Résolvez l'équation $a * b = e$ (le neutre) d'inconnue $b$. 1 pt.
II∼4) $a$ est pour cette question la suite constante égale à 1. Résolvez l'équation $a * b = r$ (la suite arithmétique $\forall n,\ r_n = n$) d'inconnue $b$. 1 pt.

III∼0) Montrez $\forall (a,\ b) \in E^2,\ \forall n \in \mathbb{N},\ \Delta(a * b) = \Delta(a) * b + a_0.T(b) = a * \Delta(b) + b_0.T(a)$. 2 pt.
III∼1) Montrez que si $a$ et $b$ sont positives et croissantes, alors $a * b$ est positive et croissante. 2 pt.
III∼2) *On se donne trois réels* $\alpha$, $\beta$ *et* $\gamma$ *et on considère l'équation « homogène »*[2] *notée* $H$ $\forall n,\ a_{n+3} + \alpha.a_{n+2} + \beta.a_{n+1} + \gamma.a_n = 0$ *d'inconnue* $a$, *et pour* $s$ *fixée dans* $E$, *on considère l'équation* $\forall n,\ u_{n+3} + \alpha.u_{n+2} + \beta.u_{n+1} + \gamma.u_n = s_n$ *d'inconnue* $u$ *(noté* $S$*).* Mettez l'équation $H$ sous la forme $\Delta^3(a) + \lambda.\Delta^2(a) + \mu.\Delta(a) + \nu.a = 0$ pour $\lambda$, $\mu$ et $\nu$ à déterminer en fonction de $\alpha$, $\beta$ et $\gamma$. 1 pt.
III∼3) $h$ est la solution de $H$ de conditions initiales $h_0 = h_1 = 0$ et $h_2 = 1$. On pose $u = h * s$. Montrez : $\Delta(u) = \Delta(h) * s$, $\Delta^2(u) = \Delta^2(h) * s$. Exprimez $\Delta^3(u) + \lambda.\Delta^2(u) + \mu.\Delta(u) + \nu.u$ uniquement à l'aide de $s$. 4 pt.

## Convolution de fonctions.

1. *« algèbre », c'est juste pour dire qu'on peut les additionner entre elles, les multiplier entre elles, et les multiplier par un réel*
2. *le physicien dit « sans second membre »*

IV∼0) *On note* $(C, +, \times, .)$ *l'ensemble des application de classe* $C^\infty$ *de* $\mathbb{R}$ *dans* $\mathbb{R}$. *Pour* $f$ *et* $g$ *dans* $C$, *on pose* $f * g = x \longmapsto \int_0^x f(x-t).g(t).dt$. Montrez que $*$ est commutative (on ne prouvera pas que$*$ est une loi interne). [1 pt.]

IV∼1) *Pour tout réel* $\lambda$, *on note* $L_\lambda$ *l'application* $x \longmapsto e^{\lambda.x}$. Explicitez $L_\lambda * L_\mu$ (en distinguant le cas $\lambda = \mu$ et le cas $\lambda \neq \mu$). [2 pt.]

IV∼2) *Pour tout entier naturel* $p$, *on note* $M_p$ *la fonction polynôme* $x \longmapsto x^p$. Montrez pour $p$ et $q$ dans $\mathbb{N}$ que $M_p * M_q$ est de la forme $\lambda_{p,q}.M_{p+q+1}$ où $\lambda_{p,q}$ sera un coefficient que vous déterminerez. [3 pt.]

V∼0) *Nous n'avons pas assez de théorèmes dans notre cours pour démontrer la résultat* $(\star) : (f * g)' = f(0).g + (f' * g) = g(0).f + (f * g')$.
Montrez quand même déjà $f(0).g + (f' * g) = g(0).f + (f * g')$. [1 pt.]

V∼1) Montrez quand $(\star)$ même si $f$ est de la forme $L_\lambda$ (et $g$ quelconque) en pensant à sortir $e^{\lambda.x}$ de l'intégrale avant de dériver comme un produit. [2 pt.]

V∼2) Montrez quand même $(\star)$ si $f$ est de la forme $M_p$ (et $g$ quelconque) en pensant à développer et sortir les $x^k$ de l'intégrale avant de dériver comme un produit. [3 pt.]

**Convolution de fonctions et équations différentielles.**

VI∼0) $a_0$ *à* $a_{n-1}$ *sont* $n$ *coefficients réels,* $s$ *est une fonction continue. On note* $H$ *et* $S$ *les deux équations différentielles* $H : (y^{(n)} + a_{n-1}.y^{(n-1)} + \ldots + a_0.y = 0)$ *et* $S : (y^{(n)} + a_{n-1}.y^{(n-1)} + \ldots + a_0.y = s)$. Montrez que si $h$ est solution de $H$ avec condition initiale $h(0) = h'(0) = \ldots = h^{(n-2)}(0) = 0$ et $h^{(n-1)}(0) = 1$, alors $h * s$ est solution de $S$. [3 pt.]

VI∼1) Donnez la solution de l'équation $y'' - 3.y' + 2.y = 0$ avec condition initiale $y_0 = 0$ et $y'_0 = 1$. [2 pt.]

VI∼2) Donnez une solution de $y''_t - 3.y'_t + 2.y_t = 2.t.e^t$. [3 pt.]

**Convolution de suites et dénombrement.**

VII∼0) *On appelle partition d'un ensemble non vide* $A$ *tout ensemble de parties de* $A$, *non vides, deux à deux disjointes, dont la réunion est égale à* $A$, *appelées cellules de la partition.* [3] *On pose* $b_0 = 1$ *et pour tout entier naturel* $n$ *strictement positif, on note* $b_n$ *le nombre de partitions de* $\{1, 2, \ldots n\}$. Calculez $b_n$ pour $n$ de 0 à 4. [2 pt.]

VII∼1) Montrez pour tout $n$ : $b_{n+1} = \sum_{k=0}^{n} \binom{n}{k}.b_k$ (on pourra classer les partitions de $\{1, 2, \ldots n+1\}$ selon le cardinal de la cellule contenant $n+1$).

VII∼2) Calculez $b_5$. [1 pt.]

VII∼3) Écrivez une procédure Python qui prend en entrée $n$ et retourne la liste des $b_k$ pour $k$ de 0 à $n$. [3 pt.]

VII∼4) On pose $a_n = \frac{b_n}{n!}$. Montrez $(n+1).a_{n+1} = (c * a)_n$ où $c$ est une suite que vous préciserez. [2 pt.]

VIII∼0) On pose $\varphi = t \longmapsto e^{e^t - 1}$. Montrez : $\varphi' = \exp \times \varphi$. [1 pt.]

VIII∼1) Montrez pour tout couple d'application $(u, v)$ de classe $C^\infty$ : $(u \times v)^{(n)} = \sum_{k=0}^{n} \binom{n}{k}.u^{(n-k)}.v^{(k)}$. [3 pt.]

VIII∼2) Déduisez : $\varphi^{(n+1)}(0) = \sum_{k=0}^{n} \binom{n}{k}.\varphi^{(k)}(0)$. [2 pt.]

VIII∼3) Montrez : $b_n = \varphi^{(n)}(0)$ pour tout $n$. [2 pt.]

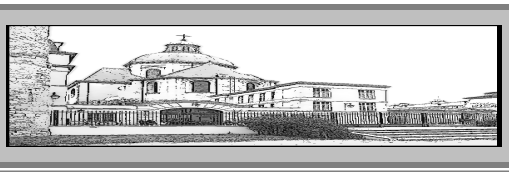
---

Colle ce Sucri sur ta copie en face de la question dont tu souhaites doubler le nombre de points.
Ou même sur la copie de ton voisin.
Ou sinon, recopie le.
Mais une seule fois.



3. *Par exemple, l'ensemble* $\{1, 2, 3\}$ *a cinq partitions :*
$\{\{1\}, \{2\}, \{3\}\}$ *composée de trois cellules*
$\{\{1, 2\}, \{3\}\}, \{\{2, 3\}, \{1\}\}$, *et* $\{\{3, 1\}, \{2\}\}$, *(composées de deux cellules)*
$\{\{1, 2, 3\}$, *composée d'une seule cellule.*

LYCEE CHARLEMAGNE
Friday the $13^{th}$
M.P.S.I.2



2024 2025
DS04
CORRECTION

DS04 720 enties à additionner.

La meilleure méthode : regrouper chaque nombre avec son « miroir », et sommer.

Il y a en effet 360 couples tels que

| | | | | |
|---|---|---|---|---|
| 123456 | 125463 | 546231 | 653241 | abcdef |
| 654321 | 652314 | 231546 | 124536 | $\alpha\beta\gamma\delta\varepsilon\varphi$ |

avec $\alpha = 7 - a$, $\beta = 7 - b$ et ainsi de suite.
On forme bien 360 couples (aucun élément n'est son propre « miroir ») et la somme de chaque couple vaut par construction 777777.
La somme cherchée vaut $360 \times 777\,777$ ce qui fait 279 999 720 (mais est moins joli que $360 \times 777\,777$ et moins explicite).

---

On crée ensuite une procédure qui détecte les nombres qui satisfont la contrainte.
Ils sont entre $10^6$ et $10^7$ (donc un `range`), et contiennent nos six chiffres et rien qu'eux.
Comment valider ce critère ? On convertit l'entier en chaine de caractère, et on fait un bon gros test.

```
S = 0
for n in range(10**6, 10**7) :
....st = string(n)
....if ('1' in st) and ('2' in st) and ('3' in st) and ('4' in st) and ('5' in st) and ('6'
in st) :
........S += n
```

*On n'est pas obligé de vérifier que chaque chiffre n'est bien présent qu'une fois.*

En effet, si l'entier à six chiffres contient le `1`, le `2`, le `3`, le `4`, le `5` et le `6`, il ne peut rien contenir d'autre, et chacun y est une seule fois.

On peut aussi convertir l'entier en chaine de caractères et même en liste de caractères.
On trie ensuite la liste, et si elle coïncide avec la liste `['1', '2', '3, '4, '5', '6']`, on valide.

```
for n in range(10**5, 10**6) :
....st = list(string(n))
....st.sort()
....if st == ['1', '2', '3, '4, '5', '6'] :
........S += n
```

L'exécution de ce programme est plus longue, car on doit trier à chaque fois une liste.

---

On écrit chaque permutation $abcdef$ sous la forme $a.10^5 + b.10^4 + c.10^3 + d.10^2 + e.10 + f$ avec pour $a$, $b$, $c$, $d$, $e$ et $f$ les six entiers dans le désordre.

On calcule donc $\sum_{\sigma\in S_6}\Big(\sum_{k=1}^{6}\sigma(k).10^{k-1}\Big) = \sum_{k=1}^{6}\Big(\sum_{\sigma\in S_6}\sigma(k)\Big).10^k$

Mais chaque somme $\Sigma_{\sigma\in S_6}\sigma(k)$ est la même. Elle est faite de 6! termes, où chaque entier de 1 à 6 intervient exactement $(6-1)!$ fois.
On somme finalement

$$\sum_{\sigma\in S_6}\Big(\sum_{k=1}^{6}\sigma(k).10^{k-1}\Big) = \sum_{k=1}^{6}\Big(\sum_{\sigma\in S_6}\sigma(k)\Big).10^{k-1} = \sum_{k=1}^{6}\Big(5!.(1+2+3+4+5+6)\Big).10^{k-1} = 5!.21.111111$$

On retrouve la même somme. Sous forme autrement factorisée.

---

Pour la somme des carrés, on refait presque de même pour la méthode informatique.

```
S = 0
for n in range(10**5, 10**6) :
....st = string(n)
....if ('1' in st) and ('2' in st) and ('3' in st) and ('4' in st) and ('5' in st) and ('6'
in st) :
........S += n*n
```

*On rappelle qu'il est plus simple de demander* `n*n` *que* `n**2`*, puisque avec* `n**2`*, Python regarde « combien vaut l'exposant ? un entier ? alors je multiplie efficacement et je trouve* `n*n` *».*

On trouve 129 158 041 750 920 et ça n'a aucun intérêt.

Et ce n'est pas le carré du précédent. J'espère ne pas trouver cette bêtise.
Sinon, pour information, ça fait $2^3 \times 3^4 \times 5 \times 7^2 \times 13 \times 37 \times 457 \times 3701$, mais je n'en vois pas l'intérêt.

---

Sinon, on part aussi de la formule $\sum_{\sigma\in S_6} \Big( \sum_{k=1}^{6} \sigma(k).10^{k-1} \Big)^2$. Ou plutôt de

$$\sum_{\sigma\in S_6} \Big( \Big( \sum_{i=1}^{6} \sigma(i).10^{i-1} \Big).\Big( \sum_{k=1}^{6} \sigma(k).10^{k-1} \Big) \Big)$$

fidèle au principe « pas un carré mais deux sommes avec deux variables différentes ».
On développe en un grand grand sigma

$$\sum_{\substack{\sigma\in S_6 \\ i\leqslant 6,\ k\leqslant 6}} \Big( \sigma(i).\sigma(k).10^{i-1}.10^{k-1} \Big)$$

On ré-indexe

$$\sum_{\substack{i\leqslant 6 \\ k\leqslant 6}} \Big( 10^{i-1}.10^{k-1}.\Big( \sum_{\sigma\in S_6} \sigma(i).\sigma(k) \Big) \Big)$$

A présent, pour $i$ et $k$ fixés, on somme des $\sigma(i).\sigma(k)$ pour les $n!$ permutations.
On doit distinguer deux cas.
Pour $i$ égal à $k$, on a une somme de carrés : $\sum_{\sigma\in S_n} (\sigma(k))^2$.
Mais qu'importe la valeur de $k$. Les nombres $\sigma(k)$ vont être tous les entiers de 1 à $n$, chacun apparaissant $(n-1)!$ fois (donc total $n!$ termes).

$$\sum_{\sigma\in S} (\sigma(k))^2 = (n-1)!.\sum_{j=1}^{n} j^2 = (n-1)!.\frac{n.(n+1).(2.n+1)}{6}$$

Application numérique ici : 120.91.

Et que trouve-t-on pour $i$ différent de $k$ ? Que donnent les $\sigma(i).\sigma(k)$ ? On trouve tous les couples $p.q$ possibles avec $p$ différent de $q$ (il y en a $\frac{n.(n-1)}{2}$), chacun intervenant avec la même probabilité (ou fréquence).
La somme donne donc $\frac{n!}{\frac{n.(n-1)}{2}}.\sum_{p\neq q} p.q$.
La somme $\sum_{p\neq q} p.q$ cache $\sum_{\substack{p\leqslant n \\ q\leqslant n}} p.q - \sum_{p=q} p.q$ et vaut $(\sum_{p} p).\Big(\sum_{q} q\Big) - \frac{n.(n+1).(2.n+1)}{6}$.
Dans le cas précis de $n = 6$ il reste 48.350.

Avec courage, on termine

$$\sum_{\substack{i\leqslant 6 \\ k\leqslant 6 \\ i\neq k}} \Big( 10^{i-1}.10^{k-1}.120.91 \Big) + \sum_{\substack{i\leqslant 6 \\ k\leqslant 6 \\ i=k}} \Big( 10^{i-1}.10^{k-1}.48.350 \Big)$$

Et on calcule trois quantités utiles

$$\sum_{\substack{i\leqslant 6 \\ k\leqslant 6 \\ i\neq k}} \Big( 10^{i-1}.10^{k-1} \Big) = \Big( \sum_{i\leqslant 6} 10^{i-1} \Big)^2 = 111111^2$$

$$\sum_{\substack{i\leqslant 6\\ k\leqslant 6\\ i=k}} \left(10^{i-1}.10^{k-1}\right) = \sum_{i\leqslant 6} 10^{2.i-2} = 10101010101$$

$$\sum_{\substack{i\leqslant 6\\ k\leqslant 6\\ i\neq k}} \left(10^{i-1}.10^{k-1}\right) = 111111^2 - 10101010101$$

et on combine le tout.

## DS04 Opérateur $T$.

On prend une suite géométrique $a$ de premier terme$a_0$ et de raison $r$. Pour tout $n$, on a $a_n = a_0.r^n$.
On se donne un entier $n$ et on calcule $(T(a))_n = a_0.r^{n+1}$ et on reconnaît une suite géométrique de premier terme $a_0.r$ et de même raison $r$.

## DS04 Opérateur $\Delta$.

Je copie et colle : On prend une suite géométrique $a$ de premier terme$a_0$ et de raison $r$. Pour tout $n$, on a $a_n = a_0.r^n$.
On se donne un entier $n$ et on calcule $(\Delta(a))_n = a_0.r^{n+1} - a_0.r^n = a_0.(r-1).r^n$et on reconnaît une suite géométrique de premier terme $a_0.(r-1)$ et de même raison $r$.

---

$k$ est fixé dans ce qui suit, entier naturel non nul. On décide de noter $(a_n)$ la suite de terme général $\binom{n}{k}$.

On calcule alors pour tout $n$ donné $(\Delta(a))_n = a_{n+1} - a_n = \binom{n+1}{k} - \binom{n}{k} = \binom{n}{k-1}$On reconnaît une suite du même type, mais pour l'entier $k$ précédent.

Et pour $k$ égal à 0, la suite $(a_n)$ est constante et sa dérivée $\Delta(a)$ est nulle.

---

On se donne $a$ et $n$. On calcule

$$(\Delta(\Delta(a)))_n = (\Delta(a))_{n+1} - (\Delta(a))_n = (a_{n+1+1} - a_{n+1}) - (a_{n+1} - a_n) = a_{n+2} - 2.a_{n+1} + a_n$$

La seule façon de comprendre ici $\Delta^2(a)$ c'est de voir $(\Delta \circ \Delta)(a)$ ou encore $\Delta(\Delta(a))$.

On recommence en appliquant encore $\Delta$

$$(\Delta(\Delta(\Delta(a))))_n = (\Delta(\Delta(a)))_{n+1} - (\Delta(\Delta(a)))_n = (a_{n+1+2} - 2.a_{n+1+1} + a_{n+1}) - (a_{n+2} - 2.a_{n+1} + a_n)$$

et on a les termes avec des coefficients binomiaux.

---

Sans se tromper d'étage, on applique $\Delta$ $k$ fois (la variable $n$ va avoir un autre rôle)

$$\forall n,\ (\Delta^k(a))_n = a_{n+k} - \binom{k}{1}.a_{n+k-1} + \binom{k}{2}.a_{n+k-2} - \dots$$

en version malpropre pour l'instant (ce sont bien des $\binom{k}{i}$ où $k$ est le nombre de fois où on dérive).
On poursuit en mettant un sigma, avec cette fois une variable de sommation qui ne peut s'appeler ni $k$ i n ; on va dire $i$. On a des binomiaux de la ligne d'indice $k$ et un clignotant qui ne dépend que de $i$

$$\forall n,\ (\Delta^k(a))_n = \sum_{i=0}^{k} (-1)^i.\binom{k}{i}.a_{n+k-i}$$

et on termine donc avec $(-1)^k.a_n$ qui semble coïncider avec ce qui précède.

Si on doit le prouver, on fait une récurrence.
Pas sur $i$, c'est une variable muette.
Pas sur $n$ car on voit mal comment passer d'un terme au suivant dans la suite $\Delta^k(a)$.
Mais $k$ semble la bonne variable, d'autant qu'on a initialisé pour deux ou trois valeurs de $k$ (et même $k = 0$, $k = 1$,

$k = 2$ et $k = 3$ en lisant bien).

*Si on doit le faire ici : on se donne k et on suppose la formule vraie pour tout n. On regarde alors au rang $k+1$*

$$\forall n,\ \Delta^{k+1}(a))_n = (\Delta(\Delta^k(a))_n = (\Delta^k(a))_{n+1} - (\Delta^k(a))_n$$

$$\forall n,\ (\Delta^{k+1}(a))_n = \sum_{i=0}^{k}(-1)^i.\binom{k}{i}.a_{n+1+k-i} - \sum_{i=0}^{k}(-1)^i.\binom{k}{i}.a_{n+k-i}$$

*On décale les indices $j = i-1$ dans la première*

$$\forall n,\ (\Delta^{k+1}(a))_n = \sum_{j=-1}^{k-1}(-1)^{j+1}.\binom{k}{j+1}.a_{n+k-j} - \sum_{i=0}^{k}(-1)^i.\binom{k}{i}.a_{n+k-i}$$

*On isole deux termes, et on fusionne les sommes*

$$\forall n,\ (\Delta^{k+1}(a))_n = 1.a_{n+k} - (-1)^k.a_n + \sum_{i=0}^{k-1}\left((-1)^{i+1}.\binom{k}{i+1}.a_{n+k-i} - (-1)^i.\binom{k}{i}.a_{n+k-i}\right)$$

*on arrange encore un peu mieux*

$$\forall n,\ (\Delta^{k+1}(a))_n = 1.a_{n+k} + (-1)^{k+1}.a_n + \sum_{i=0}^{k-1}(-1)^{i+1}.\left(\binom{k}{i+1} + \binom{k}{i}\right).a_{n+k-i}$$

*on fait appel à la formule de Pascal*

$$\forall n,\ (\Delta^{k+1}(a))_n = 1.a_{n+k} + (-1)^{k+1}.a_n + \sum_{i=0}^{k-1}(-1)^{i+1}.\binom{k+1}{i+1}.a_{n+k-1}$$

*on ré-indexe*

$$\forall n,\ (\Delta^{k+1}(a))_n = 1.a_{n+k} + (-1)^{k+1}.a_n + \sum_{j=1}^{k}(-1)^{j}.\binom{k+1}{j}.a_{n+k-j+1}$$

*et enfin on recolle les deux termes qui manquent (sachant $\binom{k+1}{0} = \binom{k+1}{k+1} = 1$*

$$\forall n,\ (\Delta^{k+1}(a))_n = \sum_{j=0}^{k+1}(-1)^{j}.\binom{k+1}{j}.a_{n+k+1-j}$$

*la formule est héréditaire, et initialisée. Elle est valable pour tout k.*

---

Mais ensuite, un élève propose (selon mon énoncé) de prendre les choses de plus haut. A l'étage des transformations.
La formule $(\Delta(a))_n = a_{n+1} - a_n = (T(a))_n - a_n = (T(a) - a)_n$ permet de dire (en libérant $n$) : $\Delta(a) = T(a) - a$ puis en libérant $a$ : $\Delta = T - Id$.

Mais alors, si on élève à la puissance $k$ , avec la formule du binôme

$$\Delta^k = (T - Id)^k = \sum_{i=0}^{k}\binom{k}{i}.(-1)^i.T^{k-i}.Id^k$$

On l'applique sur une suite $a$

$$\forall a,\ \Delta^k(a) = \Big(\sum_{i=0}^{k}\binom{k}{i}.(-1)^k.T^{k-i}.Id^k\Big)(a) = \sum_{i=0}^{k}\binom{k}{i}.(-1)^k.T^{k-i}(a)$$

On l'applique ensuite à un rang $n$

$$\forall a,\ \forall n,\ (\Delta^k(a))_n = \sum_{i=0}^{k}\binom{k}{i}.(-1)^k.(T^{k-i}(a))_n = \sum_{i=0}^{k}\binom{k}{i}.(-1)^k.a)_{n+k-i}$$

Reste à justifier l'usage de la formule du binôme sur une opération qui n'est pas une multiplication, mais une composition.

Ici, on compose des applications linéaires, et la composition est distributive sur l'addition.

Et on a besoin de commutativité.

Ici : $T \circ Id = Id \circ T$.

Bref, finalement, c'est légitime, mais pas aussi simple que d'écrire des choses très formelles comme ci dessus.

## DS04 Produit de convolution.

Quand on effectue le produit de convolution de deux suites $a$ et $b$ on obtient une nouvelle suite $a * b$.

On connait son terme généra $(a*b)_n = \sum_{k=0}^{n} a_{n-k}.b_k$.

Mais si on échange les rôles de $a$ et $b$, on obtient a priori une nouvelle suite de terme général $(b*a)_n = \sum_{k=0}^{n} b_{n-k}.a_k$.

Mais on prouve l'égalité terme à terme (c'est à dire $\forall n$) par renversement

$$(a*b)_n = \sum_{k=0}^{n} a_{n-k}.b_k = \sum_{p=0}^{n} a_p.b_{n-p} = (b*a)_n$$

Si on n'arrive pas à manipuler les sigma, on voit la symétrie

$$a_n.b_0 + a_{n-1}.b_1 + a_{n-2}.b_2 + \ldots + a_1.b_{n-1} + a_0.b_n$$

---

On se donne à présent trois suites $a$, $b$ et $c$,et pour être prudent, on nomme les objets : $a*b = \beta$, $(a*b)*c = \gamma$.
On se donne $n$ quelconque et on calcule le terme général de $\gamma$

$$\gamma_n = \sum_{k=0}^{n} \beta_k.c_{n-k} = \sum_{k=0}^{n} \Big( \sum_{i=0}^{k} a_i.b_{k-i} \Big).c_{n-k}$$

Pas pratique. En fait, tout passe mieux avec des sommes par condition

$$\beta_p = (a*b)_p = \sum_{i+j=p} a_i.b_j$$

$$\gamma_n = \sum_{p+k=n} \beta_p.c_k = \sum_{p+k=n} \sum_{i+j=p} a_i.b_j.c_k = \sum_{i+j+k=n} a_i.b_j.c_k$$

Le calcul pour $a*(b*c)$ aboutit à la même chose.

## DS04 Procédure Python.

On dispose donc de deux listes `a` et `b` de même longueur qu'on va appeler `n` et on doit construire une nouvelle liste de même longueur.
On va donc l'initialiser à `0` partout.
Et ensuite, on va parcourir les cases une par une

```
def convolution(a, b) : #list of float, list of float -> list of float
....N = len(a)
....c = [0]*N
....for n in range(N) :
........calcul de c[n]
```

Chaque `c[n]` est une somme, ici déjà initialisée à `0`.
Pour effectuer la somme, on va prendre une variable `k` comme dans la formule, qui va aller de `0` à `n` (inclus).

```
def convolution(a, b) : #list of float, list of float -> list of float
....N = len(a)
....c = [0]*N
....for n in range(N) :
........for k in range(n+1) :
```

On applique juste la formule du joli sigma

```
def convolution(a, b) : #list of float, list of float -> list of float
....N = len(a)
....c = [0]*N
....for n in range(N) :
........for k in range(n+1) :
............c += a[n-k]*b[k]
....return c
```

On peut aussi construire `c` peu à peu

```
def convolution(a, b) : #list of float, list of float -> list of float
....N = len(a)
....c = [ ]
....for n in range(N) :
........s = 0
........for k in range(n+1) :
............s += a[n-k]*b[k]
........c.append(s)
....return c
```

## DS04 Elément neutre.

On prend pour $a$ la suite $(1,\ 0,\ 0\ldots)$ qu'on note donc $e$ : $e_0 = 1$ et $e_n = 0$ pour $n \geqslant 1$.

On se donne une suite quelconque $b$ et on calcule le terme général de $e * b$

$$(e*b)_n = \sum_{k=0}^{n} e_k.b_{n-k} = e_0.b_n + \sum_{k=1}^{n} e_k.b_{n-k} = 1.b_n + 0 = b_n$$

L'égalité terme à terme donne $e * b = b$. La suite $a$ est bien neutre. A droite comme à gauche puisque la loi est commutative.

Et pour une loi interne, le neutre est unique.

## DS04 Equation convolutive.

On prend donc la suite constante égale à 1 dans le rôle de $a$ et une suite inconnue dans le rôle de $b$.
On veut que la suite « produit » $a * b$ soit égale à la suite neutre $(1,\ 0,\ 0,\ 0,\ldots)$.
On écrit les équations une à une :

$$\begin{array}{llllll} 1.b_0 & & & & = & 1 \\ 1.b_0 & +1.b_1 & & & = & 0 \\ 1.b_0 & +1.b_1 & +1.b_2 & & = & 0 \\ 1.b_0 & +1.b_1 & +1.b_2 & +1.b_3 & = & 0 \end{array}$$

et ainsi de suite.

On trouve $b_0 = 1$, $b_1 = -1$, $b_2 = 0$, $b_{3=0}$ et ainsi de suite. Bref, à l'étage des suites :

$$(1,\ 1,\ 1,\ 1,\ 1,\ldots) * (1,\ -1,\ 0,\ 0,\ 0,\ldots) = (1,\ 0,\ 0,\ 0,\ldots)$$

Mais trois équations ne suffisent pas, il faut de la rigueur.
On vérifie déjà $a_0.b_0 = 1$, c'est bon pour le premier terme de la suite.
On se donne ensuite $n$ plus grand que 1 et on calcule

$$(a*b)_n = \sum_{k=0}^{n} a_{n-k}.b_k = a_n.b_0 + a_{n-1}.b_1 + \sum_{k=2}^{n} a_{n-k}.b_k = 1.1 + 1.(-1) + 0 = 0 = e_n$$

Et résolvons maintenant l'équation $a * c = r$ d'inconnue $c$ et de second membre $r$, la suite arithmétique $(0,\ 1,\ 2,\ 3,\ldots)$.
Sans écrire de système, on n'en a plus besoin !
On a trouvé une suite $b$ vérifiant $b * a = e$ (le neutre). On part donc de $a * c = r$ et on compose à gauche par $b$

(symétrique) : $b * (a * c) = b * r$.
Par associativité, symétrie et neutralité, il reste $c = b * r$, et à l'étage des suites

$$(c_0,\ c_1,\ c_2,\ c_3, \ldots) = (1,\ -1,\ 0,\ 0,\ 0,\ 0, \ldots) * (0,\ 1,\ 2,\ 3, \ldots)$$

et on calcule au fur et à mesure

$$\begin{array}{rclllcl} c_0 & = & 1.0 & & & = & 0 \\ c_1 & = & 1.1 & -1.0 & & = & 1 \\ c_2 & = & 1.2 & -1.1 & +0 & = & 1 \\ c_3 & = & 1.3 & -1.2 & +0 & = & 1 \\ c_4 & = & 1.4 & -1.3 & +0 & = & 1 \end{array}$$

et ainsi de suite.

Oui, bon, il était facile de résoudre le système et de voir que la suite $(0,\ 1,\ 1,\ 1, \ldots)$ était la solution.
Mais c'était moins stylé.

> *Il s'agissait de deux questions sans aucune difficulté.*
> *Hormis une énorme difficulté : comprendre la question posée, car la question n'est pas « calculez cette intégrale », mais « posez vous les bonnes questions, qui cherche-t-on (non, pas n, mais une suite), que doit on vérifier ». Et c'est le type de question qui teste votre intelligence mathématique face à votre simple capacité calculatoire. Bref, sup et plus terminale. Et ça tue certains élèves.*

## DS04 "Intégration par parties".

On se donne à nouveau $a$ et $b$, on se donne $n$ et on calcule $(\Delta(a * b))_n$ c'est à dire $(a * b)_{n+1} - (a * b)_n$ avec pour objectif de trouver $(\Delta(a) * b)_n$ plus un terme. Oui, il faut regarder la suite terme à terme, avec donc un joli $\forall n$

$$\forall n,\ (\Delta(a * b))_n = (a * b)_{n+1} - (a * b)_n$$

$$\forall n,\ (\Delta(a * b))_n = \sum_{k=0}^{n+1} a_{n+1-k}.b_k - \sum_{k=0}^{n} a_{n-k}.b_k$$

On isole un terme et on fusionne les deux sommes de mêmes indices

$$\forall n,\ (\Delta(a * b))_n = a_0.b_{n+1} + \sum_{k=0}^{n} \left( a_{n+1-k}.b_k - a_{n-k}.b_k \right)$$

On prend les choses d'un peu plus haut

$$\forall n,\ (\Delta(a * b))_n = a_0.(T(b))_n + \sum_{k=0}^{n} (\Delta(a))_{n-k}.b_k$$

On libère $n$ en identifiant

$$(\Delta(a * b)) = a_0.T(b) + (\Delta(a)) * b$$

Pour l'autre égalité, on dit que les rôles sont symétriques, et on intervertit donc $a$ et $b$

$$(\Delta(b * a)) = b_0.T(a) + (\Delta(b)) * a$$

## DS04 Positivité et croissance.

On se donne $a$ et $b$ deux suites positives et croissantes ($\forall n \in \mathbb{N},\ a_{n+1} \geqslant a_n \geqslant 0$).
On définit alors la suite $a * b$. On vérifie directement sa positivité.
Pour tout $n$, $(a * b)_n$ est une somme de $n+1$ termes positifs : $\sum_{k=0}^{n} a_{n-k}.b_k \geqslant 0$.
Pour la croissance, on doit étudier le signe de $(a * b)_{n+1} - (a * b)_n$ pour $n$ quelconque donné.
Surtout, on ne tape pas sur les hypothèses pour essayer d'arriver au résultat, c'est comme à chaque fois la mauvaise méthode.

Mais qu'est ce que sur $(a * b)_{n+1} - (a * b)_n$ ? C'est notre accroissement $(\Delta(a * b))_n$ calculé au dessus.
Et on le remplace par une somme de deux termes : $((\Delta(a)) * b)_n$ et $a_0.b_{n+1}$.
Le produit $a_0.b_{n+1}$ est positif comme produit de deux réels positifs.
Et $\Delta(a)$ est une suite positive ($(a_n)$ croissante) de même que $b$ (hypothèse). Leur convolée est positive.

C'est fini. En enchainant simplement les questions, comme un étudiant de Prépas, et non en reprenant tout à zéro comme un collégien passant le bac.

## DS04 Suite récurrente linéaire.

On cherche le rapport entre deux formules

$$\forall n,\ a_{n+3}+\alpha.a_{n+2}+\beta.a_{n+1}+\gamma.a_n=0$$

et $T^3(a)+\lambda.\Delta^2(a)+\mu.\Delta(a)+\nu.a=0$ que j'écris

$$\forall n,\ (\Delta^3(a))_n+\lambda.(\Delta^2(a))_n+\mu.(\Delta(a))_n+\nu.a_n=0$$

et même

$$\forall n,\ \begin{array}{cccc} a_{n+3} & & & \\ -3.a_{n+2} & +\lambda.a_{n+2} & & \\ +3.a_{n+1} & -2.\lambda.a_{n+1} & +\mu.a_{n+1} & \\ -a_n & +\lambda.a_n & -\mu.a_n & +\nu.a_n \end{array} =0$$

Les deux équations sont identiques si et seulement si on a $\begin{array}{cccccc} -3 & +\lambda & & = & \alpha \\ 3 & -2.\lambda & +\mu & = & \beta \\ -1 & +\lambda & -\mu & +\nu = & \gamma \end{array}$.

Si $\alpha$, $\beta$ et $\gamma$ sont donnés, on trouve $\lambda$, $\mu$ et $\nu$ $\begin{array}{lllll} \lambda & = & 3 & +\alpha & \\ \mu & = & 3 & +2.\alpha & +\beta \\ \nu & = & 1 & +\alpha & +\beta \quad +\gamma \end{array}$

Si $\lambda$, $\mu$ et $\nu$ sont donnés, on trouve $\alpha$, $\beta$ et $\gamma$.
On a les deux sens d'implication.

---

On suppose donc que $h$ est solution de $H$ avec conditions indiquées.

On définit alors la suite $u=h*s$ c'est à dire $u_n=\sum_{k=0}^{n} h_{n-k}.s_k$ si on en a besoin (mais en aura-t-on besoin ?).

On calcule alors puisque c'est demandé

$$\Delta(u)=\Delta(h*s)=\Delta(h)*s+h_0.T(s)=\Delta(h)*s$$

puisque $h_0$ est nul. Et on recommence

$$\Delta^2(u)=\Delta(\Delta(h)*s)=\Delta(\Delta(h))*s+(\Delta(h))_0.T(s)=\Delta^2(h)*s$$

cette fois, on a calculé $(\Delta(h))_0=h_1-h_0=0$. On rempile

$$\Delta^3(u)=\Delta(\Delta^2(h)*s)=\Delta(\Delta^2(h))*s+(\Delta^2(h))_0.T(s)=\Delta^3(h)*s+T(s)$$

puisque cette fois $(\Delta^2(h))_0=h_2-2.h_1+h_0=1$.

Si on somme avec les coefficient, on a alors

$$\alpha.\Delta^3(u)+\lambda.\Delta^2(u)+\mu.\Delta(u)+\nu.u=\Delta^3(h)*s+T(s)+\lambda.\Delta^2(h)*s+\mu.\Delta(h)*s+\nu.h*s$$

On regroupe en une seule convolution et un terme à part

$$\alpha.\Delta^3(u)+\lambda.\Delta^2(u)+\mu.\Delta(u)+\nu.u=T(s)+\Big(\Delta^3(h)+\lambda.\Delta^2(h)+\mu.\Delta(h)+\nu.h\Big)*s$$

Or, le contenu de la grande parenthèse est nul car $h$ est solution homogène.
Il reste

$$\alpha.\Delta^3(u)+\lambda.\Delta^2(u)+\mu.\Delta(u)+\nu.u=T(s)$$

A une petite translation près, $u=h*s$ est solution de l'équation « avec second membre ».
Il faudrait reprendre avec $u=h*s'$ et $s'$ un antécédent de $s$ par $T$.

*Bref, on sait résoudre l'équation même quand il y a un second membre grâce aux convolutions.*

## DS04 Commutativité de la convolution de fonctions.

On se donne deux fonctions $f$ et $g$ et on doit comparer point par point $f * g$ et g*f .
Pour tout $x$, on a

$$(f*g)(x) = \int_{t=0}^{x} f(x-t).g(t).dt \text{ et } (g*f)(x) = \int_{u=0}^{x} g(x-u).f(u).du$$

et c'est volontairement que j'ai appelé une fois la variable muette $t$ et une autre fois $u$ (après tout, c'est une variable muette).
Il suffit ensuite d'effectuer le changement de variable $u = x - t$ pour passer de l'une à l'autre.

## DS04 Convolution et exponentielles.

On se donne $\lambda$ et $\mu$ (distincts), puis on se donne $x$ et on calcule

$$(L_\lambda * L_\mu)(x) = \int_0^x e^{\lambda.(x-t)}.e^{\mu.t}.d = e^{\lambda.x}.\int_0^x e^{(\mu-\lambda).t}.dt = e^{\lambda.x}.\Big[\frac{e^{(\mu-\lambda).t}}{\mu-\lambda}\Big]_{t=0}^{x}$$

On estime en 0 et en $x$ et on multiplie par $e^{\lambda.x}$

$$(L_\lambda * L_\mu)(x) = \frac{e^{\mu.x} - e^{\lambda.x}}{\mu-\lambda}$$

et en libérant $x$ pour vivre à l'étage des fonctions

$$L_\lambda * L_\mu = \frac{L_\mu - L_\lambda}{\mu-\lambda}$$

En revanche, si $\lambda$ est égal à $\mu$, certaines exponentielles disparaissent

$$(L_\lambda * L_\lambda)(x) = \int_0^x e^{\lambda.(x-t)}.e^{\lambda.t}.d = e^{\lambda.x}.\int_0^x dt = x.e^{\lambda.x}$$

Cette fois, en libérant $x$ : $L_\lambda * L_\lambda = Id.L_\lambda$.

*Et si vous pensez aux solutions critiques dans le cas d'une racine double pour une équation différentielle, vous avez raison.*

## DS04 Convolution et polynômes.

On se donne $p$ et $q$ (par symétrie des rôles, on va supposer $p \leqslant q$), et on calcule pour $x$ quelconque l'intégrale $\int_0^x (x-t)^p.t^q.dt$ par exemple par parties. On fait baisser l'exposant de $(x-t)$ et monter celui de $t^q$

| | | |
|---|---|---|
| $(x-t)^p$ | $\hookrightarrow$ | $-p.(x-t)^{p-1}$ |
| $t^q$ | $\hookleftarrow$ | $t^{q+1}/(q+1)$ |

En 0, c'est $t^{q+1}$ qui est nul. En $x$, c'est $(x-t)^p$ qui est nul

$$\int_0^x (x-t)^p.t^q.dt = \Big[\frac{1}{q+1}.(x-t)^p.t^{q+1}\Big]_0^1 + \frac{p}{q+1}.\int_0^x (x-t)^{p-1}.t^{q+1}.dt = \frac{p}{q+1}.\int_0^x (x-t)^{p-1}.t^{q+1}.dt$$

On recommence, sachant que $p$ a diminué d'une unité et $q$ a augmenté d'une. Le terme crochet est encore nul

$$\int_0^x (x-t)^p.t^q.dt = \frac{p}{q+1}.\int_0^x (x-t)^{p-1}.t^{q+1}.dt = 0 + \frac{p.(p-1)}{(q+1).(q+2)}.\int_0^x (x-t)^{p-2}.t^{q+2}.dt$$

Par récurrence sur le nombre $k$ d'intégrations par parties

$$(M_p * M_q)(x) = \int_0^x (x-t)^p.t^q.dt = \frac{p.(p-1)\dots(p-i+1)}{(q+1).(q+2)\dots(q+i)}.\int_0^x (x-t)^{p-i}.t^{q+i}.dt$$

Une fois qu'on a épuisé $p$ (arrivé à $i = p$), il reste

$$(M_p * M_q)(x) = \frac{p.(p-1)\dots(p-p+1)}{(q+1).(q+2)\dots(q+p)}.\int_0^x (x-t)^0.t^{q+p}.dt$$

On intègre en $\dfrac{t^{p+q+1}}{p+q+1}$ et on trouve

$$(M_p * M_q)(x) = \frac{p.(p-1)\dots(p-p+1)}{(q+1).(q+2)\dots(q+p)}.\frac{x^{p+q+1}}{(p+q+1)}$$

En libérant $x$, on a, à l'étage des fonctions $M_p * M_q = \lambda_{p,q}.M_{p+q+1}$ avec $\boxed{\lambda_{p,q} = \dfrac{p.(p-1)\dots(p-p+1)}{(q+1).(q+2)\dots(q+p).(p+q+1)}}$

(un terme de plus au dénominateur qu'au numérateur).

On peut ensuite reconnaître $\dfrac{p!}{(p+q+1)!/q!} = \dfrac{p!.q!}{(p+q+1)!}$ et même $\dfrac{1}{(p+q+1).\binom{p+q}{p}} = \dfrac{1}{(p+q+1).\binom{p+q}{q}}$.

## DS04 Dérivation et convolution.

On doit prouver $(f' * g) + f_0.g = g_0.f + (f * g')$. On se donne un $x$ et on compare

$$f(0).g(x) + \int_0^x f'(t).g(x-t).dt \text{ et } f(x).g(0) + \int_0^x f(t).g'(x-t).dt$$

La différence vaut

$$f(0).g(x) - f(x).g(0) + \int_0^x \Big(f'(t).g(x-t) - f(t).g'(x-t)\Big).dt$$

Or, le contenu de l'intégrale n'est autre que $\Big(t \longmapsto f(t).g(x-t)\Big)$ (un signe moins quand on dérive le deuxième terme, car on compose $t \longmapsto x-t \longmapsto g(x-t)$). La différence vaut donc

$$f(0).g(x) - f(x).g(0) + \Big[f(t).g(x-t)\Big]_{t=0}^{x}$$

et ceci fait bien M .

*On pouvait aussi partir de $f(0).g(x) + \int_0^x f'(t).g(x-t).dt$ et intégrer par parties*

| | | |
|---|---|---|
| $f'(t)$ | $\hookleftarrow$ | $f(t)$ |
| $g(x-t)$ | $\hookrightarrow$ | $-g'(x-t)$ |

*A vous de comprendre que c'est deux fois la même méthode.*

---

On se donne à présent $g$ quelconque, et $f$ égale à $L_\lambda$. On va dériver $x \longmapsto \int_0^x e^{\lambda.(x-t)}.g(t).dt$ qu'on écrit

$$x \longmapsto e^{\lambda.x}.\int_0^x e^{-\lambda.t}.g(t).dt$$

et même $x \longmapsto e^{\lambda.x}.G(x)$ où $G$ est la primitive de $t \longmapsto e^{-\lambda.t}.g(t)$.
On dérive alors ce produit

$$\Big(x \longmapsto e^{\lambda.x}.\int_0^x e^{-\lambda.t}.g(t).dt\Big)' = \Big(x \longmapsto \lambda.e^{\lambda.x}.\int_0^x e^{-\lambda.t}.g(t).dt + e^{\lambda.x}.e^{-\lambda.x}.g(x)\Big)$$

On simplifie ce qu'on peut

$$\Big(x \longmapsto e^{\lambda.x}.\int_0^x e^{-\lambda.t}.g(t).dt\Big)' = \Big(x \longmapsto \int_0^x \lambda.e^{\lambda(x-.)t}.g(t).dt + 1.g(x)\Big)$$

et on a bien $(L_\lambda)' * g$ dans l'intégrale et $L_\lambda(0).g(x)$ dans l'autre terme. C'est ce qu'on voulait.

*Difficulté : on a appris à dériver $x \longmapsto \int_0^x h(t).dt$ en $x \longmapsto h(x)$.*

*Ce qu'il faut éviter c'est la salade de variable où on écrit $\Big(x \longmapsto \int_0^x h(t).dt\Big)' = (x \longmapsto h(t))$ qui n'a aucun sens.*

*Et le premier qui me dit « mais on ne m'a jamais dit ça », je lui demande « tu calculez $\int_0^x h(t).dt = H(x) - H(0)$ où $H$ est une primitive de $h$ ; c'est quoi une primitive ? » ; et l'élève de répondre : $H' = h$. Tiens c'est exactement ce qu'on a fait ci dessus.*

A présent, $g$ est encore quelconque, et dans le rôle de $f$ on prend le polynôme $X^p$ (ou plutôt a fonction polynôme).
On va dériver $x \longmapsto \int_0^x (x-t)^p.g(t).dt$.
$x$ est à la fois borne de l'intégrale mais aussi sous le signe somme. Comment faire ? On développe par la formule du binôme et on sépare par linéarité

$$(M_p * g)(x) = \sum_{k=0}^{p} \binom{p}{k}.x^k.\int_0^x (-t)^{p-k}.g(t).dt$$

Cette fois, on peut dériver la somme puis chaque produit

$$(M_p * g)'(x) = \sum_{k=0}^{p} \binom{p}{k}.\Big(k.x^{k-1}.\int_0^x (-t)^{p-k}.g(t).dt + x^k.(-x)^{p-k}.g(x)\Big)$$

On peut séparer en deux sommes

$$(M_p * g)'(x) = \Big(\sum_{k=0}^{p} \binom{p}{k}.k.x^{k-1}.\int_0^x (-t)^{p-k}.g(t).dt\Big) + \sum_{k=0}^{p} \binom{p}{k}.x^k.(-x)^{p-k}.g(x)$$

Dans le première, le terme d'indice $k = 0$ est nul, ne l'écrivons pas.
Quant au produit $\binom{p}{k}.k$ il vaut $p.\binom{p-1}{k-1}$ (formule comité président).
Sinon, $\sum_{k=0}^{p} \binom{p}{k}.x^k.(-x)^{p-k}$ vaut $(x-x)^p$, ce qui ne fait pas grand chose (sauf pour $p = 0$). A ce stade

$$(M_p * g)'(x) = \Big(\sum_{k=1}^{p} p.\binom{p-1}{k-1}.x^{k-1}.\int_0^x (-t)^{p-k}.g(t).dt\Big) + (x-x)^p.g(x)$$

Sortons le $p$ de la somme et rentrons les intégrales en une seule

$$(M_p * g)'(x) = p.\int_0^x \Big(\sum_{k=1}^{p} \binom{p-1}{k-1}.x^{k-1}(-t)^{p-k\prime}\Big).g(t).dt + (0)^p.g(x)$$

Quitte à changer $k-1$ en $k'$, la quantité dans la somme est $(x-t)^{p-1}$. On a bien obtenu

$$(M_p * g)'(x) = \int_0^x p.(x-t)^{p-1}.g(t).dt + (0)^p.g(x)$$

On a bien obtenu $(M_p * g)'(x) = ((M_p') * g)(x) + M_p(0).g(x)$ et en effaçant le $\forall x$, on a bien
$\boxed{(M_p * g)' = ((M_p') * g) + M_p(0).g}$

## DS04 Equation différentielle et convolution.

On part donc de $h(0) = h'(0) = \ldots = h^{(n-2)}(0) = 0$ et $h^{(n-1)}(0) = 1$ et $h^{(n)} + a_{n-1}.h^{(n-1)} + \ldots + a_0.h = 0$.

On pose donc $f = h * s$ (c'est $x \longmapsto \int_0^x h(x-t).s(t).dt$ mais on ne va plus utiliser cette forme, mais on va préférer utiliser ce qu'on a montré avant).
On doit calculer $f'$, $f''$ et ainsi de suite, jusqu'à $f^{(n)}$ et reporter dans l'équation $S$ (avec second membre).

Profitons du travail précédent :

$$f' = (h * s)' = h(0).s + h' * s = 0 + h' * s$$

en profitant de la condition initiale sur $h$. On recommence

$$f'' = (h' * s)' = h'(0).s + (h')' * s = h'' * s$$

Et si on recommence on a de la même façon

$$f^{(3)} = (h'' * s)' = h''(0).s + (h'')' * s = h^{(3)} * s$$

jusqu'à

$$f^{(n-1)} = (h^{(n-2)} * s)' = h^{(n-2)}(0).s + h^{(n-1)} * s = h^{(n-1)} * s$$

et enfin, légèrement à part :

$$f^{(n)} = (h^{(n-1)} * s)' = h^{(n-1)}(0).s + h^{(n-1+1)} * s = s + h^{(n)} * s$$

On met des coefficients et on somme

$$f^{(n)} + a_{n-1}.f^{(n-1)} + \ldots + a_0.f = (s + h^{(n)} * s) + a_{n-1}.h^{(n-1)} * s + \ldots + a_0.h * s$$

Mais on peut regrouper en un seul terme (distributivité de la convolution)

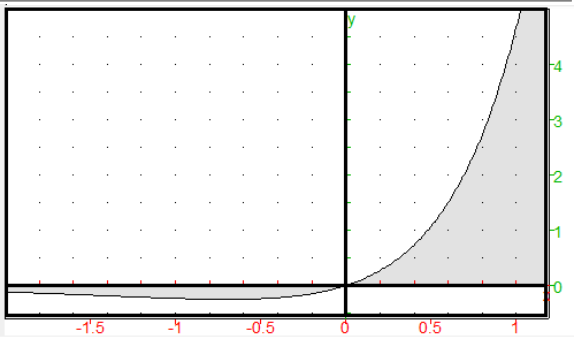
$$f^{(n)} + a_{n-1}.f^{(n-1)} + \ldots + a_0.f = s + \left(h^{(n)} * + a_{n-1}.h^{(n-1)} + \ldots + a_0.h\right) * s$$

Mais le contenu de la parenthèse est nul, car $h$ est solution de $H$. Il reste donc

$$f^{(n)} + a_{n-1}.f^{(n-1)} + \ldots + a_0.f = s$$

et on reconnaît que $f$ est solution de l'équation « avec second membre ».

## DS04 Cas particulier.




Pour résoudre l'équation différentielle $y'' - 3.y' + 2.y = 0$, on applique le cours.
Équation caractéristique : $\lambda^2 - 3.\lambda + 2 = 0$.
Spectre : $\{1, 2\}$.
Solutions : $t \longmapsto a.e^t + b.e^{2.t}$ avec $a$ et $b$ dépendant des conditions initiales.

On impose deux conditions initiales : $y_0 = 0$ et $y'_0 = 1$. On trouve $a + b = 0$ et $a + 2.b = 1$.
La solution est donc $t \longmapsto -e^t + e^{2.t}$, valable sur $\mathbb{R}$.

Et pour l'équation avec second membre (non homogène).
On utilise justement ce qui a été fait au dessus.
On appelle $s$ l'application $t \longmapsto 2.t.e^t$ et $h$ notre solution homogène.
On sait alors que l'application $h * s$ est solution de l'équation de second membre $s$.

Il n'y a plus qu'à calculer ce produit de convolution :

$$x \longmapsto \int_0^x (e^{2.(x-t)} - e^{x-t}).(2.t.e^t).dt$$

On développe et on sort $e^x$

$$x \longmapsto 2.e^{2.x}.\int_0^x t.e^{-t}.dt - 2.e^x.\int_0^x t.dt$$

On calcule la première intégrale par parties et la seconde sans effort $x \longmapsto 2.e^{2.x} - (x^2 - 2.x - 2).e^x$

On peut évidemment proposer et vérifier. Mais auriez vous deviné cette forme ?
Avec le cours spécifique sur les équations différentielles, vous apprendrez à le faire.
Mais avec des second membres encore plus tordus, vous aurez quand même par convolution une forme explicite (mais intégrale).

## DS04 Partitions.

On commence par dénombrer :

| $0 \to 1$ | $1 \to 1$ | $2 \to 2$ | $3 \to 5$ | $4 \to 15$ |
|---|---|---|---|---|
| | $\{\{1\}\}$ | $\{\{1,2\}\}$ | $\{\{1,2,3\}\}$ | $\{\{1,2,3,4\}\}$ |
| | | | $\{\{1,2\}, \{3\}\}$ | $\{\{1,2,3\}, \{4\}\}$ |
| | | $\{\{1\}, \{2\}\}$ | $\{\{2,3\}, \{1\}\}$ | $\{\{1,2,4\}, \{3\}\}$ |
| | | | $\{\{3,1\}, \{2\}\}$ | $\{\{1,3,4\}, \{2\}\}$ |
| | | | | $\{\{2,3,4\}, \{1\}\}$ |
| | | | | $\{\{1,2\}, \{3,4\}\}$ |
| | | | $\{\{1\}, \{2\}, \{3\}\}$ | $\{\{1,3\}, \{2,4\}\}$ |
| | | | | $\{\{1,4\}, \{2,3\}\}$ |
| | | | | $\{\{1,2\}, \{3\}, \{4\}\}$ $\{\{1,3\}, \{2\}, \{4\}\}$ |
| | | | | $\{\{1,4\}, \{2\}, \{3\}\}$ $\{\{2,3\}, \{1\}, \{4\}\}$ |
| | | | | $\{\{2,4\}, \{1\}, \{3\}\}$ $\{\{3,4\}, \{1\}, \{2\}\}$ |
| | | | | $\{\{1\}, \{2\}\ \{3\}, \{4\}\}$ |

La formule pour $b_4$ doit être cohérente avec la suite : $b_4 = \sum_{k=0}^{3} \binom{3}{k}.b_k = 1.1 + 3.1 + 3.2 + 1.5 = 15$.

---

On considère qu'on connaît les $b_k$ pour $k$ de 0 à $n$.
On regarde maintenant les partitions de l'ensemble $\{1,\ 2, \ldots n+1\}$ par disjonction de cas (dès lors, on va sommer les possibilités).
Si on prend une partition $P$ faite de quelques cellules, l'élément $n+1$ est dans une et une seule des cellules.

En effet, la réunion des cellules donne $\{1,\ 2,\ \ldots n+1\}$, et les intersections deux à deux sont vides.

La cellule contenant $n+1$ a alors un cardinal qu'on va noter $k$, compris entre 0 et $n$.

> *Pour k égal à 0, $n+1$ est seul dans sa cellule et il reste une partition de $\{1,\ 2, \ldots n\}$.*
> *Pour k égal à n, la partition est monocellulaire tout le monde est avec $n+1$.*

Pour former cette cellule, on choisit $k$ éléments parmi $n$.
Ensuite, il reste une partition des $n-k$ éléments qui restent.

La formule obtenue est $\sum_{k=0}^{n} \binom{n}{k}.b_{n-k}$ et, en renversant la somme $\sum_{i=0}^{n} \binom{n}{i}.b_i$. Il ne reste qu'à rappeler $\binom{n}{i} = \binom{n}{n-i}$.

Pour saisir, avec $n = 3$. On regarde la taille de la cellule contenant le « nouvel » élément 3

| cellule de taille 1 | cellule de taille 2 | | | cellule de taille 3 | | |
|---|---|---|---|---|---|---|
| donc seul | avec 1 | avec 2 | avec 3 | avec 1 et 2 | avec 1 et 3 | avec 2 et 3 |
| $\{\{1,2,3\}, \{4\}\}$ | $\{\{2\}, \{3\}, \{1,4\}\}$ | $\{\{1\}, \{3\}, \{2,4\}\}$ | $\{\{1\}, \{2\}, \{3,4\}\}$ | $\{\{3\}, \{1,2,4\}\}$ | $\{\{2\}, \{1,3,4\}\}$ | $\{\{1\}, \{2,3,4\}\}$ |
| $\{\{1,2\}, \{3\}, \{4\}\}$ | $\{\{2,3\}, \{1,4\}\}$ | $\{\{1,3\}, \{2,4\}\}$ | $\{\{1,2\}, \{3,4\}\}$ | | | |
| $\{\{1,3\}, \{2\}, \{4\}\}$ | | | | | | |
| $\{\{2,3\}, \{1\}, \{4\}\}$ | | | | cellule de taille 4 | | |
| $\{\{1\}, \{2\}, \{3\}, \{4\}\}$ | | | | $\{\{1,2,3,4\}\}$ | | |
| | | | | | | |

Cette formule étant établie, on l'utilise pour calculer $b_5$ (un point vite gagné, à condition de savoir appliquer une définition, bref, un teste de votre capacité comme futur ingénieur)

$$b_5 = \sum_{k=0}^{4} \binom{4}{k}.b_k = 1.0 + 5.1 + 10.2 + 5.5 + 1.15 = 52$$

Bon, on ne va pas en donner la liste ici.
Peut être les différentes formes et le décompte pour chacune.

| $\{\{a,b,c,d,e\}\}$ | $\{\{a\}, \{b,c,d,e\}\}$ | $\{\{a\}, \{b\}, \{c,d,e\}\}$ | $\{\{a,b\}, \{c,d,e\}\}$ |
|---|---|---|---|
| 1 | 5 | 10 | 10 |

| | $\{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}\}$ | $\{\{a\}, \{b\}, \{c\}, \{d,e\}\}$ | $\{\{a\}, \{b\}, \{c\}, \{d,e\}\}$ |
|---|---|---|---|
| | 1 | 5 | 10 |

La procédure va construire de proche en proche les b à partir de cette formule.
On estime, suivant son niveau de compétences, qu'on a une fonction pour calculer des binomiaux

```
from math import *
```
mais quelle déchéance !

```
def facto(n) :
....p = 1
....for k in range(1, n+1) :
........p *= k
....return p
```

```
def binom(n, k) :
....p = 1
....for i in range(k) :
........p = p*(n-i) // (i+1)
....return p
```

```
def binom(n, k) :
....return facto(n) //
(facto(k)*facto(n-k))
```

```
def binom(n, k) :
....if k == 0 :
........return 1
....return (n-k)*binom(n, k-1) //
(k+1)
```

Ensuite, on crée une liste dont le seul élément est le 1, puis on avance pas à pas en créant une somme par boucle.

```
def partition(N) :
....P = [1]
....for n in range(N) :
........s = 0
........for k in range(n+1) :
............s += binom(n, k)*P[k]
........P.append(s)
....return P
```

Application numérique à l'ordinateur :
[1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, 190899322, 1382958545, 10480142147, 82864869804, 682076806159, 5832742205057, 51724158235372]

---

On remplace les binomiaux par leur définition dans la formule de récurrence forte

$$b_{n+1} = \sum_{k=0}^{n} \frac{n!}{k!.(n-k)!}.b_k$$

puis les $b_k$ par leur définition à partir des $a_k$

$$(n+1)!.a_{n+1} = \sum_{k=0}^{n} \frac{n!}{k!.(n-k)!}.k!.b_k = \sum_{k=0}^{n} \frac{n!}{(n-k)!}.a_k$$

On simplifie par $n!$

$$(n+1).a_{n+1} = \sum_{k=0}^{n} \frac{1}{(n-k)!}.a_k$$

On reconnaît qu'on a convolé la suite $(a_p)$ avec la suite $\left(\frac{1}{p!}\right)$.

## DS04 Fonction annexe $\varphi$.

On gagne un point en dérivant

$$\left(x \longmapsto e^{e^x-1}\right)' = \left(x \longmapsto e^x.e^{e^x} - 1\right)$$

et on l'écrit à l'étage des fonctions $\varphi' = \exp \times \varphi$.

---

La formule de Leibniz est un classique, qui fait partie du cours de Prépas. Elle se démontre par récurrence sur $n$.
La formule de Leibniz à l'ordre 0 dit juste « si $a$ et $b$ existent, alors $a \times b$ existe et on a

$$a \times b = (a \times b)^{(0)} = 1.a^{(0)} \times b^{(0)}$$

Supposons pour un entier $n$ donné quelconque la formule vraie à l'ordre $n$, et ajoutons l'hypothèse : $a$ et $b$ sont $n+1$ fois dérivables.
Chaque terme de la somme $\sum_{k=0}^{n} \binom{n}{k} \times a^{(n-k)} \times b^{(k)}$ est dérivable au oins une fois de plus.
En effet, $(a^{(n-k)} \times b^{(k)})' = (a^{(n-k)})' \times b^{(k)} + a^{(n-k)} \times (b^{(k)})'$ et $a^{(n-k)}$ et $b^{(k)}$ sont bien dérivables au moins une fois de plus.
En utilisant la linéarité de la dérivation, on a donc

$$((a \times b)^{(n)})' = \sum_{k=0}^{n} \binom{n}{k} \times \left(a^{(n-k+1)} \times b^{(k)} + a^{(n-k)} \times b^{(k+1)}\right)$$

si l'on se fie au calcul précédent.

On sépare en deux sommes

$$(a \times b)^{(n+1)} = \sum_{k=0}^{n} \binom{n}{k} \times a^{(n-k+1)} \times b^{(k)} + \sum_{k=0}^{n} \binom{n}{k} \times a^{(n-k)} \times b^{(k+1)}$$

On décale dans la deuxième somme en posant $k' = k+1$

$$(a \times b)^{(n+1)} = \sum_{k=0}^{n} \binom{n}{k} \times a^{(n-k+1)} \times b^{(k)} + \sum_{k'=1}^{n+1} \binom{n}{k'-1} \times a^{(n+1-k')} \times b^{(k')}$$

Les variables étant muettes, on peut fusionner les deux sommes en une au moins sur la partie commune

$$(a \times b)^{(n+1)} = \binom{n}{0} \times a^{(n+1)} \times b^{(0)} + \sum_{k=1}^{n} \left(\binom{n}{k} + \binom{n}{k-1}\right) \times a^{(n-k+1)} \times b^{(k)} + \binom{n}{n} \times a^{(0')} \times b^{(n+1)}$$

La partie centrale donne $\binom{n+1}{k} \times a^{(n+1-k)} \times b^{(k)}$ et les deux termes du bout peuvent s'écrire
$\binom{n+1}{0} \times a^{(n+1)} \times b^{(0)}$ et $\binom{n+1}{n+1} \times a^{(0)} \times b^{(n+1)}$ et compléter la somme.

---

On repart de $\varphi' = \exp \times \varphi$ et on dérive $n$ fois de chaque côté

$$\varphi^{(n+1)} = (\varphi')^{(n)} = \left(\exp \times \varphi\right)^{(n)} = \sum_{k=0}^{n} \binom{n}{k} . \exp^{(n-k)} . \varphi^{(k)}$$

On rappelle que la fonction exponentielle se dérive en elle même.
Il ne reste plus qu'à calculer en 0

$$\varphi^{(n+1)}(0) = \sum_{k=0}^{n} \binom{n}{k} . \exp(0) . \varphi^{(k)}(0)$$

Attention. On prouve la formule par récurrence, puis on applique en $x = 0$.
En revanche, la formule $\varphi^{(n+1)}(0) = \sum_{k=0}^{n} \binom{n}{k} . \exp(0) . \varphi^{(k)}(0)$ ne peut pas faire l'objet d'une récurrence, car on ne peut plus dériver. Il n'y a plus de variable par rapport à laquelle dériver.

---

Les deux suites $(b_n)$ et $(\varphi^{(n)}(0))$ vérifient la même relation de récurrence forte. Et la même condition initiale en $= 0$.
Elles sont donc égales.
Proprement, on montre $b_n = \varphi^{(n)}(0)$ pour tout $n$ par récurrence forte.

On initialise avec $b_0 = 1$ (convention) et $\varphi(0) = 1$ aussi (là, si vous tenez à voir une convention, c'est $\varphi^{(0)} = \varphi$).

On se donne un entier $n$ et on suppose que pour tout $k$ de 0 à $n$ on a $b_k = \varphi^{(k)}(0)$.
On calcule $b_{n+1}$ par la formule $b_{n+1} = \sum_{k=0}^{n} \binom{n}{k} . b_k$.
On remplace par hypothèse de récurrence : $b_{n+1} = \sum_{k=0}^{n} \binom{n}{k} . \varphi^{(k)}(0)$.
On reconnaît la formule issue de la formule de Leibniz :

$$b_{n+1} = \sum_{k=0}^{n} \binom{n}{k} . b_k = \sum_{k=0}^{n} \binom{n}{k} . \varphi^{(k)}(0) = \varphi^{(n+1)}(0)$$

*L'exercice pouvait continuer (sujet de concours Institut de l'Université de Paris 2009).*