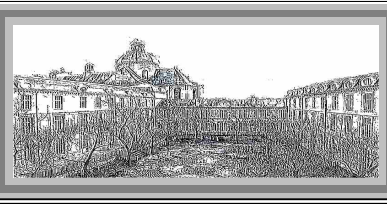
LYCEE CHARLEMAGNE
Lundi 13 mai
M.P.S.I.2



2023 2024

TD28

∘0∘ Trouvez un nombre qui est soixante dix sept fois plus grand que le chiffre de ses unités.

On vaut $100.a + 10.b + c = 77.c$ en écrivant le nombre $\overline{abc}$.
$76.c$ doit être multiple de10. C'est que c vaut 5 (ou 0, solution trop facile !).
La solution est $\boxed{385}$ et c'est plié.

On pouvait aussi chercher tout de suite parmi les multiples de 77.

∘1∘ Le problème que Pavl Halmos aimait poser à ses étudiants : les concombres sont composés de quatre vingt dix neuf pour cent d'eau (si si, et nous, c'est plus que quatre vingt dix pour cent). Le Cours des Halles en bas de chez moi en a acheté cinq cent kilos. Mais après le week-end, ils ne sont plus formés que de quatre vingt dix huit pour cent d'eau. Combien de kilos lui reste-t-il à vendre ?

Les concombres sont constitués de matière non aqueuse et d'eau. Le pourcentage indiqué nous informe d'un rapport de masse : $\frac{masse\ d'eau}{masse\ totale} = \frac{99}{100}$. Connaissant la masse totale, on a la masse d'eau : 99×500 kilogrammes.
Et par soustraction, le un pour cent de la masse, c'est la matière non aqueuse : un pour cent des cinq cent kilos. Il a cinq kilos de matière qui ne soit pas de l'eau.

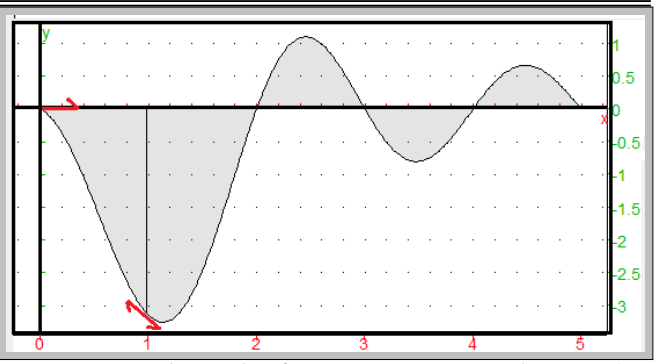
A la fin du week-end, il lui reste toujours ces cinq kilos de matière, et de l'eau. Combien ? Je ne sais pas, disons une masse m. Et on a cette fois : $\frac{m}{m+5} = \frac{98}{100}$. On effectue : $m = 245$. Putain de perte !
Ce qu'il lui reste est un stock de

5 kilos de matière non aqueuse	245 kilos d'eau	250 kilos de concombre

la moitié du stock s'est évaporée. Les concombres ont perdu la moitié de leur eau, et cela a juste fait passer un pourcentage de 99% à 98% ...

Hors sujet : *Ah, bientôt le retour de la saison des concombres, on va pouvoir remanger des légumes et fruits gorgés d'eau : concombres, tomates, melon, pêches...*
En attendant, on reste sur les légumes de saison (courges, cardes, raves...). Et les fruits de saison (pommes de garde...).

∘2∘ ♡ J'ai l'impression que $x \longmapsto \frac{\sin(\pi.x)}{\ln(x)}$ se prolonge par continuité en 0 et en 1. Prouvez le. Et donne moi l'équation de la tangente en 0 puis en 1.



quand x tend vers 0, le numérateur reste borné, et le dénominateur tend vers l'infini. Le quotient tend vers 0 (on irait jusqu'à le qualifier de forme surdéterminée, du type « zéro sur l'infini »).

En 1, numérateur et dénominateur tendent vers 0. Forme indéterminée.
Mais on va poser $x = 1 + h$ et effectuer des développements limites : $\frac{\sin(\pi + \pi.h)}{\ln(1+h)} = \frac{-\sin(\pi.h)}{\ln(1+h)}$ par simple calcul déjà.
Ensuite, on développe vraiment ou on passe aux équivalents : $\frac{\sin(\pi + \pi.h)}{\ln(1+h)} \sim \frac{-\pi.h}{h}$. Le quotient tend vers $-\pi$.
On veut un développement limite d'ordre 1 du quotient.

On veut la dérivée en 1 par limite des taux d'accroissement :

$$\frac{\dfrac{\sin(\pi.(1+h))}{\ln(1+h)}+\pi}{h}=\frac{\sin(\pi+\pi.h)+\pi.\ln(1+h)}{h.\ln(1+h)}$$

Numérateur et dénominateur tendent vers 0, mais à quelle vitesse ? Le dénominateur est équivalent à h^2.
Pour le numérateur, on développe :

$$\sin(\pi+\pi.h)=-\sin(\pi.h)=-\pi.h+\frac{\pi^3.h^3}{6}+o(h^3)$$

et $\pi.\ln(1+h)=\pi.\Big(h-\dfrac{h^2}{2}+o(h^2)\Big)$.

Le quotient est équivalent à $\dfrac{-\dfrac{\pi.h^2}{2}+o(h^2)}{h^2+o(h^2)}$ et il tend vers $\dfrac{-\pi}{2}$.

On pourra poser $f'(1)=-\pi/2$.

L'équation de la tangente en 1 est $\boxed{y=-\pi-\dfrac{\pi}{2}.(x-1)}$

◦3◦

♠ Soit f une application uniformément continue de Q dans $\mathbb{R}$. On veut montrer qu'elle se prolonge en une application continue de $\mathbb{R}$ dans $\mathbb{R}$ "par densité de $\mathbb{Q}$".
On se donne un irrationnel a. Montrez qu'il existe une suite (r_n) de rationnels qui converge vers a. Montrez que les suites (r_n) et $(f(r_n))$ sont de Cauchy et convergent donc. L'élève A veut alors poser $f(a)=\lim\limits_{n\mapsto+\infty} f(r_n)$.
L'élève B l'arrête en indiquant : mais si tu avais pris une autre suite (c_n), qui dit que tu aurais obtenu la même limite $f(a)=\lim\limits_{n\mapsto+\infty} f(c_n)$ *($f(a)$ dépendrait il du choix de la suite choisie ?)*. Que pensez vous de son objection ? Et que pensez vous de l'élève qui explique alors $(r_0,\ c_0,\ r_1,\ c_1,\ r_2,\ldots)$?
Montrez que f ainsi définie est à son tour uniformément continue.

On va prolonger f par continuité (uniforme) en a irrationnel.
Chaque intervalle $[a-2^{-n},\ a+2^{-n}]$ contient au moins un rationnel car non réduit à un point.

Rappel : tout intervalle non réduit à un point contient au moins un rationnel.
Et par petite translation, tout intervalle non réduit à un point contient au moins un irrationnel.
Et par itération : tout intervalle non réduit à un point contient une infinité de rationnels.
tout intervalle non réduit à un point contient une infinité d'irrationnels.

On en prend un dans chacun de ces intervalles.
On a une suite de rationnels (r_n) vérifiant $|a-r_n|\leqslant 2^{-n}$ pour tout n.
Par encadrement, c'est une suite de rationnels, et elle converge vers a.

Sinon, on prend $\Big(\Big[\dfrac{a}{10^n}\Big].10^n\Big)$ qui est la suite classique des approximations décimales par défaut.

La suite (r_n) est de Cauchy car convergente.

Classique : $|r_p-r_q|=|r_p-a+a-r_q|\leqslant|r_p-a|+|a-r_q|\leqslant\dfrac{\varepsilon}{2}+\dfrac{\varepsilon}{2}$ pour p et q plus grands que $N_{\varepsilon/2}$.

Il est temps d'utiliser la continuité uniforme. L'image d'une suite de Cauchy par une application uniformément continue est une suite de Cauchy.

$\boxed{\mathbb{H}}$	$\forall\varepsilon>0,\ \exists K_\varepsilon,\ \forall(p,q),\ (p\geqslant K_\varepsilon\ et\ q\geqslant K_\varepsilon)\Rightarrow(\lvert a_p-a_q\rvert\leqslant\varepsilon)$
$\boxed{\mathbb{H}}$	$\forall\varepsilon>0,\ \exists\mu_\varepsilon,\ \forall(x,y),\ (\lvert y-x\rvert\leqslant\mu_\varepsilon)\Rightarrow(\lvert f(y)-f(x)\rvert\leqslant\varepsilon)$
$\boxed{?}$	$\forall\varepsilon>0,\ \exists Q_\varepsilon,\ \forall(p,q),\ (p\geqslant Q_\varepsilon\ et\ q\geqslant Q_\varepsilon)\Rightarrow(\lvert f(a_p)-f(a_q)\rvert\leqslant\varepsilon)$

Pour ε donné, on prend $Q_\varepsilon=K_{(\mu_\varepsilon)}$ et c'est fini.

La suite $(f(r_n))$ est une suite de Cauchy dans $\mathbb{R}$.
Mais $\mathbb{R}$ est complet (résultat du cours). C'est à dire qu'il y a équivalence entre « suite de Cauchy » et « suite convergente ».

La suite $f(r_n))$ converge dans $\mathbb{R}$.
Le théorème de complétude de $\mathbb{R}$ ne dit pas vers quoi.
C'est ça la force des suites de Cauchy : on montre qu'elles convergent sans être obligé de dire vers quoi.
C'est aussi ce en quoi le physicien dira que c'est un peu inutile au premier abord...

Pourquoi ne peut on se permettre de poser $f(a) = \lim_{n\to+\infty} f(r_n)$ et dire « f est maintenant prolongée par continuité en a » ?

Le problème est qu'il il existe une suite (r_n) qui converge vers a et telle que la suite $(f(r_n))$ converge vers $f(a)$.
Et la définition est « pour toute suite (c_n) qui converge vers a, la suite $(f(c_n))$ converge vers $f(a)$ ».

On doit passer de « une suite » à « toute suite ».

Un début. Si une suite (c_n) converge vers a alors la suite (c_n) est de Cauchy
la suite $(f(c_n))$ est de Cauchy
la suite $(f(c_n))$ converge
Mais vers quoi ? Pas forcément vers la même limite que $(f(r_n))$ de tout à l'heure. Oui, il ne faut pas aller trop vite.

Il faut se convaincre que $(f(r_n))$ et $(f(c_n))$ ont la même limite. [1]

Et là, la belle astuce est entrelarder les deux suites en une seule.
On crée la suite (x_p) dont un terme sur deux vient de (c_n) et un terme sur deux de (r_n).
En l'occurrence : $x_{2.p} = r_p$ et $x_{2.p+1} = c_p$.
Les deux sous-suites $(x_{2.p})$ et $(x_{2.p+1})$ convergent vers a par définition.
Par recouvrement, la suite (x_n) converge vers a.
Elle est donc de Cauchy.
La suite image $(f(x_n))$ est de Cauchy.
La suite image $(f(x_n))$ converge.
Ses deux sous-suites $(f(x_{2.p}))$ et $(f(x_{2.p+1})$ convergent donc vers une même limite.
L'une convergeait vers la valeur qu'on avait donné $f(a)$.
L'autre convergeait vers $\lim_{n\to+\infty} f(c_n)$.
On a donc $f(c_n)$ qui converge vers $f(a)$.

f est continue en a.

On pouvait le faire en tout point a.

Partout on a pu prolonger f par continuité en tout point a irrationnel.

f est maintenant continue partout.

> *Une application uniformément continue de $\mathbb{Q}$ dans $\mathbb{R}$ se prolonge automatiquement d'une façon unique en une application continue de $\mathbb{R}$ dans $\mathbb{R}$.*
> *Plus généralement, une application uniformément continue de A dans $\mathbb{R}$ (avec A une partie dense de E, espace complet) se prolonge en une application continue de E dans $\mathbb{R}$.*
> *Ça sert à quoi ?*
> *On définit l'application $f \longmapsto \int_a^b f(t).dt$ quand f est une fonction en escalier.*
> *L'opérateur est uniformément continu.*
> *On peut le prolonger à un opérateur $f \longmapsto \int_a^b f(t).dt$ quand f est une limite de fonctions en escalier...*
> *On a prolongé la notion d'intégrale par densité des fonctions en escalier au sein de l'ensemble de toutes les fonctions...*

◦4◦ Montrez que pour tout a, $\int_0^a \frac{Atan(a.t)}{t}.dt$ existe (on la note $F(a)$).
Montrez : $F'(a) = 2.\frac{Arctan(a^2)}{a}$ *(si si, il y a bien un 2, et l'astuce constera à changer de variable pour qu'il n'y ait plus de a dans la fonction intégrée, juste dans les bornes).*

1. et il faut vous convaincre aussi, mais il y a tant parmi vous qui disent « bah, à quoi bon, il suffit de faire confiance, je ne vois pas où est le problème »

Qu'est ce qui pourrait empêcher cette intégrale d'exister. Je sais, pour vous, tout existe, du moment que vous l'écrivez. Et qu'il n'y a pas un dénominateur nul.
Tiens, et là justement, le dénominateur s'annule en 0. Et 0 est dans notre intervalle d'étude (juste au bord, mais il y est).
Toutefois, l'application $x \longmapsto \dfrac{Arctan(a.x)}{x}$ se prolonge par continuité en 0 par la valeur a (équivalents).
On a juste l'intégrale d'une application continue sur un segment, elle existe.

L'idée trop naïve est de dériver une intégrale fonction de la borne supérieure comme $a \longmapsto \int_0^a f(t).dt$. Le cours nous dit que si f est continue, on obtient $f(a)$.

Le con lui, nous dit $f(a) - f(0)$ car il n'a rien compris.

Et ce résultat se justifie en écrivant $\int_0^a f(t).dt = F(a) - F(0)$ puis en dérivant.

Mais ici, l'application sous le signe somme dépend de a. On a donc une forme

$$\int_0^a f_a(t).dt = F_a(a) - F_a(0).$$

Et il devient plus délicat de dériver.

Certains vont aller chercher des gros théorèmes de seconde année avec

$$\frac{d}{da}\Big(\int_0^a f_a(t).dt\Big) = F_a(a) + \int_0^a \frac{\partial}{\partial a}\Big(f_a(t)\Big).dt$$

C'est bien d'avoir des lectures saines.

Mais ici, il y a plus simple, et toujours pratique : un petit changement de variable pour « sortir a ».

$$\int_{t=0}^{a} \frac{Atan(a.t)}{t}.dt = \int_{u=0}^{u=a^2} \frac{Arctan(u)}{u/a}.\frac{du}{a} = \int_{u=0}^{u=a^2} \frac{Arctan(u)}{u}.du$$

On dérive maintenant une composée : $u \underset{2.a}{\longrightarrow} a^2 \underset{f(a^2)}{\longrightarrow} \int_0^{a^2} f(t).dt$.
On trouve finalement $2.a.\dfrac{Arctan(a^2)}{a^2}$ c'est à dire effectivement $2.\dfrac{Arctan(a^2)}{a}$

Et tout ça, sans avoir calculé de primitive pour la fonction sous le signe somme, une fois de plus.

Encore un exercice sans difficulté majeure (je n'ai pas dit sans difficulté), mais qui permet de voir si vous avez bien compris, ou si vous manipulez des formules sans voir ce qu'il y a derrière. Un exercice type de Sup.

∘5∘

♡ Prolongez la aux bornes du domaine : $x \longmapsto \ln(x).\ln(1-x)$. Trouvez son maximum, donnez son axe de symétrie.
Admet elle un développement limité en 0 ?

On exige $x > 0$ et $1 - x > 0$. On trouve $x \in]0,\ 1[$.
Quand x tend vers 0, on a une forme indéterminée.
On utilise un équivalent pour ce qu'on peut : $\ln(1-x) \sim_{x\to 0} -x$ et rien pour l'autre.
$\ln(1-x).\ln(x) \sim -x.\ln(x)$ c'est une forme indéterminée classique, qui tend vers 0.
Étant équivalente à une quantité de limite nulle, la fonction tend vers 0 en 0.

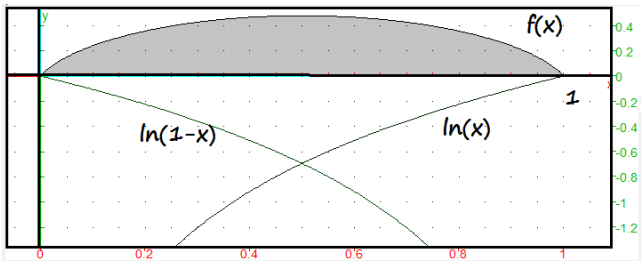
En 1, en posant $x = 1 - h$, on a le même modèle d'équivalent : $\ln(1-h).\ln(h) \sim -h.\ln(h)$.
La fonction tend aussi vers 0 en 1 (et on le sait avec l'axe de symétrie).

Oui, l'axe de symétrie est visible rien qu'avec l'écriture : $f(1-x) = f(x)$ (si elle s'appelle f).
S'il n'y avait pas le 1 ce serait de la parité $h(-t) = h(t)$.
Le 1 n'y change pas grand chose. On décale : $h(t) = f\Big(t + \dfrac{1}{2}\Big)$ (et $\dfrac{1}{2}$ est la valeur qui donne $1 - x = x$).

On a alors $h(-t) = h(t)$ puisque $f\left(-t+\frac{1}{2}\right) = f\left(1-\left(t+\frac{1}{2}\right)\right) = f\left(t+\frac{1}{2}\right)$.
h est paire et possède donc un axe de symétrie, f possède donc aussi un axe de symétrie.
On peut dire aussi que $1-x$ et x sont symétriques par rapport à $\frac{1}{2}$ (leur milieu est $\frac{1}{2}$), et ont la même image.

Rappel :

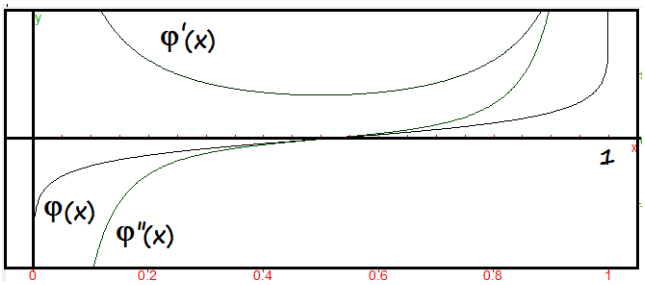
Axe de symétrie en a :	$f(2.a-x) = f(x)$
Centre de symétrie en $(a,\ b)$:	$f(2.a-x) + f(x) = 2.b$



Vu sa symétrie, son maximum doit être en $\frac{1}{2}$.
Mais après tout, elle pourrait avoir un maximum de chaque côté de $\frac{1}{2}$.
On dérive : $t \longmapsto \frac{\ln(1-t)}{t} - \frac{\ln(t)}{1-t}$ et on va étudier le signe du numérateur *(dénominateur de signe constant sur* $]0,\ 1[$*)*.

On crée donc $\varphi = x \longmapsto (1-x).\ln(x) - x.\ln(1-x)$
On dérive : $\varphi' = x \longmapsto (-\ln(x) - \ln(1-x) + \frac{1-x}{x} - \frac{x}{1-x})$.
Et pas qu'un peu, mon n'veu : $\varphi'' = x \longmapsto \left(\frac{1-2.x}{x^2.(1-x)^2}\right)$

φ' est croissante, décroissante, avec maximum en $\frac{1}{2}$ de valeur $2.\ln(2) - 2$ (négative).



φ est donc décroissante. Nulle en $\frac{1}{2}$. Elle s'annule et change de signe en $\frac{1}{2}$ (passant du positif au négatif).
f' est donc positive, puis négative. f admet son maximum en $\frac{1}{2}$.

Ce type de petit raisonnement avec tableau de variations et dérivées doit être un classique pour vous, sans aucune difficulté.
La seule subtilité est de ne garder que le numérateur de f (nomme φ)
et d'accepter de se dire qu'il faudra peut être dériver plusieurs fois.

Pour ce qui est du développement limité en 0, on peut certes s'occuper de $\ln(1-x)$. Mais il reste $\ln(x)$ qui est rétif à tout développement limité.

Et si on regarde le graphe on se doute de quelquechose. On a une demi tangente verticale en 0.
Et ceci interdit tout développement limité (pou ceux qui en ont besoin DL d'ordre n implique DL d'ordre 1 implique dérivable).

On vérifie donc : $\frac{f(x)-f(0)}{x-0} = \frac{\ln(1-x)}{x}.\ln(x)$. Le terme $\frac{\ln(1-x)}{x}$ tend vers -1 *(équivalent du numérateur)* et $\ln(x)$ tend vers l'infini.
Demi-tangente verticale. C'est fini. Le seul développement limité sera d'ordre 0.

∘6∘

Écrivez un script `Python` qui pour une permutation `sigma` donnée (sous forme de liste des images) déterminez le plus long cycle.
Par exemple, pour `[0, 3, 8, 1, 9, 6, 4, 10, 2, 7, 5]` il répondra `[4, 9, 7, 10, 5, 6]` (ou `[5, 6, 4, 9, 7, 10]`) puisque c'est le même.
Comme votre programme devra travailler sur des permutations très longues, il doit être optimisé pour avoir tous les points, et ne pas perdre de temps.

On crée déjà la petite procédure qui pour un élément donné détermine son cycle dans la permutation

```
def cycle_extrait(a, sigma) :
....L = [a]
....while not(sigma[a] in L) :
.........a=sigma[a]
.........L.append(a)
....return L
```

```
def cycle_extrait(a, sigma) :
....aa, L = a, [a]
....while not(sigma[aa] in L) :
.........a=sigma[aa]
.........L.append(aa)
....return L
```

Attention, ce script modifie la valeur de a en cours de programme. Certes, en sortant de la procédure, a reprend sa valeur, mais ce n'est as très poli. On préfèrera donc le script de droite avec variable de sauvegarde.
Ensuite, on parcourt les valeurs de a, on regarde le cycle, et on le compare au plus grand cycle déjà trouvé *(on aura initialisé au cycle de longueur 0, facile à dépasser)*

```
def plus_grand_cycle(sigma) :
....LMax =[]
....LongMax=0
....for a in range(len(sigma)) :
........cycle_a = cycle_extrait(a, sigma)
........Long = len(cycle_a)
........if Long > LongMax :
............LongMax = Long
............LMax = cycle_a
....return LMax
```

Ce script est évidemment un peu brutal, puisque on explore le cycle de chaque élément. Mais plusieurs éléments ont le même cycle. Par exemple pour `[0, 3, 8, 1, 9, 6, 4, 10, 2, 7, 5]`

qui s'écrit en fait $\overrightarrow{(0)} \circ \overrightarrow{(1\ 3)} \circ \overrightarrow{(2\ 8)} \circ \overrightarrow{(4\ 9\ 7\ 10\ 5\ 6)}$, on sort les listes suivantes :

$a=0 : [0]$	$a=1 : [1,3]$	$a=2 : [2,8]$	$a=3 : [3,1]$	$a=4 : [4,9,7,10,5,6]$	$a=5 : [5,6,4,9,7,10]$

et ainsi de suite. Le cycle déjà croisé pour 4 est revisité pour 5, 6 et les autres.
On pourra donc créer une liste des éléments déjà visités, ou marquer dans une liste les positions déjà visitées (ici, ce sera le rôle de la liste de booléens `vierge`) :

```
def plus_long-cycle(sigma) :
....LongMax=0
....LMax = []
....vierge = [True for k in range(len(sigma))] #tout est vierge a priori
....for a in range(len(sigma)) : #on va parcourir la liste
........if vierge[a] : #mais on ne prendra que les éléments jamais explorés
............vierge[a] = False
............cycle_a = [a]
............while not(sigma[a] in cycle_a) : #le cycle déjà créée
................a = sigma[a]
................vierge[a] = False #on marque qu'on est passé
................cycle_a.append(a)
............if len(cycle_a) > LongMax :
................LongMax = len(cycle_a)
................LMax = cycle_a
....return LMax
```

∘7∘

a	Pourquoi est il vrai que si f est continue à droite et continue à gauche en a, alors elle est continue en a.	.
b	Pourquoi est il vrai que si f est dérivable à droite et dérivable à gauche en a, alors pourquoi elle n'est pas forcément dérivable en a.	
c	Pourquoi est il vrai que si f est dérivable à droite et dérivable à gauche en a, alors elle est continue en a.	
d	Un $\mathcal{E}$ tracé par Enis est il homéomorphe à un $\mathcal{E}$ tracé par Ines ?	
e	Vrai ou faux : $x \longmapsto \ln(\lvert x\rvert)$ a pour dérivée $x \longmapsto \dfrac{1}{\lvert x\rvert}$?	
f	Vrai ou faux : f est dérivable en a si et seulement si $x \longmapsto f(x+a)$ est dérivable en 0.	
g	Vrai ou faux : si f est dérivable en a de dérivée $f'(a)$, alors $x \longmapsto f(-x)$ est dérivable en $-a$ de dérivée $f'(a)$.	
h	Vrai ou faux : si $x \longmapsto Arctan(f(x))$ est dérivable en a alors f est dérivable en a.	
i	Vrai ou faux : si $x \longmapsto (f(x))^2$ est dérivable en a alors f est dérivable en a.	
j	Vrai ou faux : si f est paire, alors f' est nulle en 0.	
et même http ://rogermansuy.fr/HX2/Derivables.html		

Homéomorphisme : application continue bijective dont la réciproque est aussi continue.

a : la limite à droite est gale à $f(a)$, la limite à gauche est gale à $f(a)$.
La limite sans condition est égale à $f(a)$.

---

b : la limite à droite des taux d'accroissements existe, de même que la limite à gauche.
Mais elles ne sont pas forcément gales.
L'exemple usuel est la valeur absolue, en 0.
Elle est dérivable à droite ($f'_d(0) = 1$) et à gauche ($f'_g(0) = -1$).
Elle n'est pas dérivable en 0.

---

c : Si elle est dérivable à droite, elle est continue à droite (un DL d'ordre 1 devient un DLd'ordre 0).
Si elle est dérivable à gauche, elle est continue à gauche.
Étant continue à droite et à gauche, elle est continue.

---

d : Julien L ; et Julien W ne sont plus élèves de MPSI2. Mais vous pouvez reprendre avec des prénoms de la classe de cette année. Et peut être même avec un même prénom porte par deux élèves...

---

e : Faux. $x \longmapsto \ln(\lvert x\rvert)$ a pour dérivée $x \longmapsto \dfrac{1}{x}$. C'est même dans le cours de Terminale.
Pour vous en convaincre.
Travaillez sur $\mathbb{R}^{-*}$ pour commencer. Vous avez $x \longmapsto \ln(-x)$ qui se dérive en $x \longmapsto \dfrac{-1}{-x}$.
Travaillez sur $\mathbb{R}^{-*}$ pour commencer. Vous avez $x \longmapsto \ln(x)$ qui se dérive en $x \longmapsto \dfrac{1}{x}$.

Vous pouvez aussi convaincre le physicien sur $\mathbb{R}^-$ en écrivant $\ln(-x) = \ln(x) + \ln(-1)$ et $\ln(-1)$ est une constante qui saute à la dérivation.

---

f : Vrai.
C'est même la clef pour passer des formules $f(a+h) = f(a) + h.f'(a) + o(h)$ (recommandées)
aux formules $f(x) = f(a) + (x-a).f'(a) + o(x-a)$ (sources d'erreurs chez trois quart des élèves les utilisant).

---

g : Si f est dérivable en a de dérivée $f'(a)$,alors $x \longmapsto f(-x)$ est dérivable en $-a$ mais sa dérivée est $-f'(-a)$.
C'est la dérivée d'une composée.
On peut aussi écrire $g(x) = f(-x)$ et obtenir $g(-a+h) = f(a-h) = f(a) - h.f'(a) + o(-h) = g(-a) - h.f'(a) + o(h)$.

---

h : Vrai. Il suffit de composer par tan,dérivable sur $]-\pi/2,\ \pi/2[$.
La simplification $\tan(Arctan(f(x)) = f(x)$ vient de la définition.

---

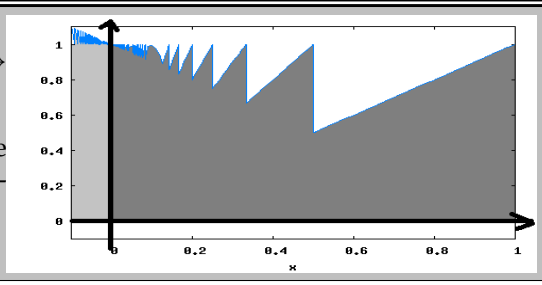
i : Si $x \longmapsto (f(x))^2$ est dérivable en a, $x \longmapsto f(x)$ n'est peut être même pas continue en a.
Prenez $x \longmapsto \left\{ \begin{array}{lll} 1 & si & x\ rationnel \\ -1 & sinon & \end{array} \right\}$.
Ici, on ne pouvait pas composer parla racine carrée. On n'a pas $\sqrt{t^2} = t$.

---

j : Si f est paire et dérivable, alors f' est impaire et donc nulle en 0.
Mais si f est paire sans être dérivable, comme la valeur absolue.
On peut aussi dire que « f paire » n'implique pas « f définie en 0 ». Alors à quoi bon dériver ?

◦8◦

♡ Donnez les points de discontinuité sur $[0,\ 1]$ de $x \longmapsto x.\left[\frac{1}{x}\right]$. Prolongez cette application par continuité en 0.
♣ Exprimez l'intégrale de cette application de 0 à 1 comme somme d'une série numérique. Donnez sa valeur, en admettant $\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$.



Messaline se fit mettre un col de loutre sur la nuque. Le Pape remercia la Duchesse de l'avoir fait mander. Il n'y a pas que dans les Postes qu'on voit de beaux Bottins. La Chine se soulève à l'appel du Japon. Frottez moi ce lard dit la charcutière, il est bien salé.
Ne cherchez pas, ce sont des parapétries de l'OuLiPo. Elles en ont la forme, l'odeur, la saveur, mais ce ne sont pas des contrepèteries.

◦9◦

Montrez que l'application $x \longrightarrow \frac{x^3+3.x}{3.x^2+1}$ est croissante sur $\mathbb{R}$. Étudiez la suite u_0 donné et pour tout n $u_{n+1} = \frac{(u_n)^3+3.u_n}{3.(u_n)^2+1}$. Discuter suivant u_0.

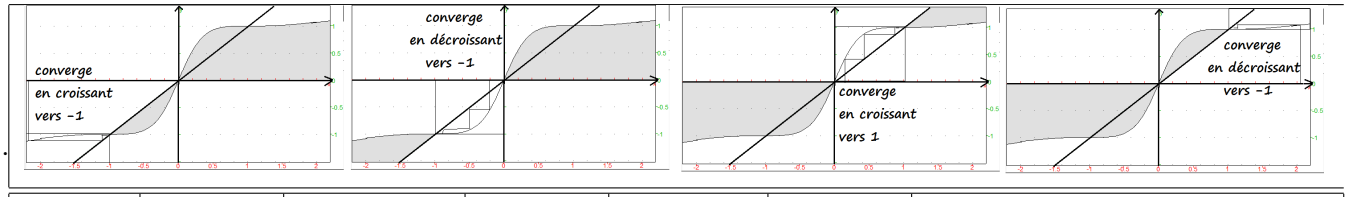
L'application est définie sur tout $\mathbb{R}$.
On n'a guère le choix, il faut la dériver et ne regarder que le numérateur :

$$(3.x^2+3).(3.x^2+1) - (x^3+3.x).(6.x) = 3.x^4 - 6.x^2 + 3 = 3.(x^2-1)^2$$

Il est positif.

Comme l'application est définie partout, la suite existe sans condition sur u_0.
L'application est croissante, la suite sera monotone (son sens de variations dépendra des deux premiers termes : si $u_0 > u_1$, alors par $f \circ f \ldots \circ f$ on a $u_{n+1} > u_n$).

Étudions quand même $f(x) - x$ non pas pour avoir tout de suite la monotonie de la suite, mais pour détecter les points fixes et découper $\mathbb{R}$ en intervalles stables.
Le numérateur est en $x.(x^2-1)$. On va découper en quatre intervalles stables.
En effet, si on a $0 < x < 1$ par exemple, alors on a $0 = f(0) < f(x) < f(1) = 1$.
Et par récurrence, si u_0 est entre 0 et 1, alors chaque u_n y est aussi.



$u_0 < -1$	$u_0 = -1$	$-1 < u_0 < 0$	$u_0 = 0$	$0 < u_0 < 1$	$u_0 = 1$	$1 < u_0$
croit	constante	décroît	constante	croît	constante	décroît
converge vers -1			vers 0	converge vers 1		

Pour la convergence, on montre comme toujours la stabilité (exemple en colonne 1 : $\forall n,\ u_n < -1$)
la monotonie (exemple en colonne 3 : $\forall n,\ u_{n+1} < u_n$)
la convergence
la valeur de la limite par élimination (exemple en colonne 7 seule limite possible 1)

1 et -1 sont stables.
0 est instable.

∘10∘ Pour tout n, on définit $f_n = t \longmapsto \dfrac{t^n - t^{2.n}}{1-t}$. Montrez que f_n se prolonge par continuité en 1. On note alors I_n l'intégrale de 0 à 1 de f_n. Montrez $I_n = \sum_{k=1}^{n} \dfrac{1}{n+k}$. Déduisez par comparaison série intégrale que I_n converge vers $\ln(2)$ quand n tend vers l'infini.

A faire.

∘11∘ On définit : $f = x \longmapsto e^{\frac{1}{\ln(x)}}$ de $]0,\ 1[$ dans $\mathbb{R}$. Prolongez la par continuité en 0 et en 1. est elle alors C^1 ? Est-ce un homéomorphisme de $[0,\ 1]$ dans $[0,\ 1]$?
Est-ce un difféomorphisme de $]0,\ 1[$ dans $]0,\ 1[$?
Est-ce un difféomorphisme de $]0,\ 1]$ dans $]0,\ 1]$?
Déterminez f^{-1}.

$f = x \longmapsto e^{\frac{1}{\ln(x)}}$ est définie sur $]0,\ 1[$. Et elle est continue, dérivable et même C^∞.

Quand x tend vers 0, $\ln(x)$ tend vers $-\infty$ et l'exponentielle tend vers 1.

Quand x tend vers 1 par valeur inférieure, son logarithme tend vers 0 (mais négatif), et $e^{\frac{1}{\ln(x)}}$ tend vers 0 (abusivement $e^{-\infty}$).

L'application est monotone, comme composée d'applications monotones :

$$x \underset{\ln}{\longrightarrow} \ln(x) \underset{\frac{1}{.}}{\longrightarrow} \frac{1}{\ln(x)} \underset{\exp}{\longrightarrow} e^{\frac{1}{\ln(x)}}$$

. L'application est décroissante.

On la dérive comme composée : $x \underset{\frac{1}{x}}{\longrightarrow} \ln(x) \underset{\frac{-1}{(\ln(x))^2}}{\longrightarrow} \frac{1}{\ln(x)} \underset{e^{\frac{1}{\ln(x)}}}{\longrightarrow} e^{\frac{1}{\ln(x)}}$ on trouve $x \longmapsto -\dfrac{e^{\frac{1}{\ln(x)}}}{x.(\ln(x))^2}$.

La dérivée est C^1 sur $]0,\ 1[$.

On va chercher si f' a une limite en 0 et/ou en 1. Pour pouvoir utiliser le théorème de la limite de la dérivée.

En 0 : $e^{\frac{1}{\ln(x)}}$ tend vers 1
$x.(\ln(x))^2$ tend vers 0 (le célèbre $x.\ln(x)$ en 0 et sa généralisation
le quotient $\dfrac{-e^{\frac{1}{\ln(x)}}}{x.(\ln(x))^2}$ tend vers $-\infty$.
La dérivée a une limite. On en déduit que f est dérivable en 0 de dérivée $-\infty$ (abus de langage) : $f'(0) = -\infty$.
Et sans abus de langage, on a une demi tangente verticale.

On vient d'utiliser le théorème de la limite de la dérivée.
Et sans lui, il fallait étudier $\dfrac{e^{\frac{1}{\ln(x)}} - 1}{x}$ *dans lequel on pouvait poser* $t = \dfrac{1}{\ln(x)}$, *qui tend vers* 0.
Le taux s'écrit alors $\dfrac{e^t - 1}{e^{\frac{1}{t}}} = \dfrac{t + o(t)}{e^{\frac{1}{t}}}$.
On change encore de variable $X = \frac{1}{t}$ *qui tend vers l'infini négatif. On a* $\dfrac{1 + o(1)}{X.e^X}$ *avec x qui tend vers* $l'infini - \infty$.
L'indéterminée du dénominateur tend vers 0, *le quotient tend vers l'infini.*
On retrouve notre demi-tangente verticale.

En 1 par valeur inférieure (ce qu'on note 1^-), on regarde encore si la dérivée a une limite. $\dfrac{-e^{\frac{1}{\ln(x)}}}{x.(\ln(x))^2}$.
Pour se faciliter la vie, on va poser $X = \dfrac{1}{\ln(x)}$ (qui tend vers $-\infty$). Cette dérivée devient $\dfrac{-e^X}{\frac{e^X}{X^2}}$ ce qui fait $-\frac{X^2.e^X}{e^X}$.
Comme X tend vers $-\infty$, c'est l'exponentielle qui l'emporte au numérateur dans la forme indéterminée. Et le dénominateur tend vers 1 . Le quotient tend vers $\dfrac{-0}{1}$ ce qui fait 0.

Le théorème de la limite de la dérivée dévoile : f est dérivable à gauche en 1 et $f'_g(1) = 0$ (demi tangente horizontale).

Résumé :

x	0	$]0,\ 1[$	1
$f'(x)$	| |	-	0
$f(x)$	1	↘	0

f est C^1 sur $]0,\ 1]$.

---

f réalise un homéomorphisme de $]0,\ 1[$ sur l'intervalle image $]0,\ 1[$ (mais dans l'autre sens).
f réalise un homéomorphisme de $]0,\ 1]$ sur l'intervalle image $[0,\ 1[$.

f réalise un homéomorphisme de $[0,\ 1[$ sur l'intervalle image $]0,\ 1]$.

f réalise un homéomorphisme de $[0,\ 1]$ sur l'intervalle image $[0,\ 1]$.

f réalise un difféomorphisme de $]0,\ 1[$ sur l'intervalle image $]0,\ 1[$.

Mais f ne réalise aucun difféomorphisme sur un des intervalles fermés, il y a toujours soit une demi-tangente verticale (f' non défini ou infini),
soit une demi tangente horizontale qui donnera une demi-tangente verticale pour f^{-1} ;

Quant à f^{-1}, c'est facile : $f(f(x)) = x$ pour tout x et $f^{-1} = f$.
ce qui se traduit par une symétrie axiale par rapport à la première bissectrice.
Quand on a un point $(x, \hat{}y)$ avec $y = f(x)$ on a aussi le point $(y,\ x)$ avec $x = f^{-1}(y) = f(y)$.

∘12∘

On définit $f = \theta \longmapsto \tan(\theta) + \dfrac{2}{2.\theta - \pi}$. Prolongez f par continuité en $\pi/2$ [a] *(on notre $\overline{f}$ l'application prolongée, juste pour avoir la rigueur emmerdante des concours)*. Montrez que $\overline{f}$ est dérivable en $\pi/2$. Montrez que $\overline{f}$ est C^1 sur $[0,\ \pi/2]$.
Calculez $\displaystyle\int_0^{\pi/2} \overline{f}(\theta).d\theta$.
On pose $I_n = \displaystyle\int_0^{\pi/2} \overline{f}(\theta).\sin(n.\theta).d\theta$, Montrez que I_n tend vers 0 quand n tend vers l'infini *(by parts)*.

a. pensez à écrire $x = \dfrac{\pi}{2} + h$ quand x doit tendre vers $\pi/2$, car c'est en 0 que vous maitrisez vos développements limités

Ayant posé $f(\theta) = \tan(\theta) + \dfrac{2}{2.\theta - \pi}$, on a l'existence en tout point de $[0, \pi/2[$ et de $]\pi/2,\ \pi]$ par exemple. Pour montrer que f se prolonge par continuité en $\pi/2$, on va regarder la limite de $f(x)$ quand x tend vers $\pi/2$.

Et surtout, on ne va pas séparer "à droite/à gauche". On n'est plus en Terminale avec des réflexes idiots. On ne sépare ainsi que quand la formule est différente à droite et à gauche...

On change de variable car c'est en 0 qu'on connaît tout : $x = \dfrac{\pi}{2} + h$ avec h qui va tendre vers 0 *(par valeur supérieure ou inférieure)* :

$$f\Big(\frac{\pi}{2} + h\Big) = \tan\Big(\frac{\pi}{2} + h\Big) + \frac{2}{\pi + 2.h - \pi} = \frac{\sin(h + \pi/2)}{\cos(h + \pi/2)} + \frac{1}{h} = \frac{-\cos(h)}{\sin(h)} + \frac{1}{h} = \frac{\sin(h) - h.\cos(h)}{h.\sin(h)}$$

On floute par développement limité connu :

$$f\Big(\frac{\pi}{2} + h\Big) = \frac{\Big(h - \dfrac{h^3}{6} + o(h^3)\Big) - h.\Big(1 - \dfrac{h^2}{2} + o(h^2)\Big)}{h.\sin(h)} = \frac{h^3/3 + o(h^4)}{h.\sin(h)}$$

(quand h tend vers 0).

On ne regarde que ce qui domine *(équivalents)* : $f\Big(\dfrac{\pi}{2} + h\Big) \sim_{h\to 0} \dfrac{h^3/3}{h.h} = \dfrac{h}{3}$.

L'équivalent tend vers 0, la quantité tend aussi vers 0 : on pose $\boxed{f(\pi/2) = 0}$

Maintenant, on regarde la dérivabilité par taux d'accroissement : $\tau = \dfrac{f(x) - \overline{f}\Big(\dfrac{\pi}{2}\Big)}{x - \dfrac{\pi}{2}}$.

On change de variable :

$$\tau = \frac{f\Big(\dfrac{\pi}{2} + h\Big) - \overline{f}\Big(\dfrac{\pi}{2}\Big)}{h} = \frac{f\Big(\dfrac{\pi}{2} + h\Big)}{h}$$

On profite de ce qui a été fait avant : $\tau \sim_{h\to 0} = \dfrac{\frac{h}{3}}{h} = \dfrac{1}{3}$.
Équivalent à un réel non nul, ce taux d'accroissement tend vers ce réel. Les taux d'accroissement ont une limite, la fonction est dérivable : $\boxed{f\left(\frac{\pi}{2}\right) = \frac{1}{3}}$

Mais on peut aussi utiliser le théorème de la limite C^1 en dérivant d'abord :

$$f' = \theta \longmapsto \frac{1}{\cos^2(\theta)} - \frac{4}{(2.x-\pi)^2}$$

Il reste à voir si f' admet une limite en $\pi/2$ et si cette limite vaut $1/2$.
On notera que si on prouve que f' admet une limite, le théorème nous donne directement que $\overline{f}$ est dérivable en $\pi/2$ de dérivée égale à ce qu'on a trouvé.
On change de variable :

$$f'\left(\frac{\pi}{2}+h\right) = \frac{1}{\sin^2(h)} - \frac{4}{(2.h)^2} = \frac{h^2-\sin^2(h)}{h^2.\sin^2(h)} = \frac{(h-\sin(h)).(h+\sin(h))}{h^2.\sin^2(h)}$$

On passe aux développements limités là où il faut :

$$h - \sin(h) = h - \left(h - \frac{h^3}{6} + o(h^3)_{h\to 0}\right) = \frac{h^3}{6} + o(h^3)_{h\to 0}$$

On passe aux équivalents $f'\left(\frac{\pi}{2}+h\right) \sim_{h\to 0} \dfrac{\frac{h^3}{6}.2.h}{h^2.h^2} = \dfrac{1}{3}$. L'équivalent devient une limite.
$\overline{f}$ est dérivable, même en $\dfrac{\pi}{2}$ et $\overline{f}'$ est C^1 *(tiens, c'est $\overline{f}'$ ou $\overline{f'}$?)*.

Pour l'intégration, on ne sépare pas. La linéarité c'est bien quand tout existe.
Mais ici, ni $\int_0^{\pi/2} \tan(\theta).d\theta$ ni $\int_0^{\pi/2} \dfrac{2}{2.\theta-\pi}.d\theta$. En revanche, on peut intégrer sur un segment $\int_0^a \left(\dfrac{\sin(\theta)}{\cos(\theta)} + \dfrac{2}{2.\theta-\pi}\right).d\theta = \Big[\ln(2.\theta-\pi) - \ln(\cos(\theta))\Big]_{\theta=0}^{\theta=a}$.
Plus proprement, on a une primitive : $\theta \longmapsto \ln\left(\dfrac{\pi-2.\theta}{\cos(\theta)}\right)$. On prend sa valeur en 0 : $\ln(\pi)$.
On prend sa limite en $\pi/2$ en posant encore $\theta = \dfrac{\pi}{2} - h$ *(on est à gauche de 0, on va prendre h positif)* : $\ln\left(\dfrac{2.h}{\sin(h)}\right)$. Le quotient tend vers 2.1. Le logarithme tend vers 0 par continuité.
Au final, il reste $\int_0^{\pi/2} \overline{f}(\theta).d\theta = \ln(2/\pi)$ *(négatif, ce qui est normal)*.

$I_n = \int_0^{\pi/2} \overline{f}(\theta).\sin(n.\theta).d\theta$ existe pour tout n. On intègre par parties en sachant que $\overline{f}$ est dérivable

$\overline{f}$	$\hookrightarrow$	$\overline{f}'$
$\sin(n.\theta)$	$\hookleftarrow$	$-\dfrac{\cos(n.\theta)}{n}$

$$I_n = \Big[\frac{\overline{f}(\theta).\cos(n.\theta)}{n}\Big]_{\theta=0}^{\theta=\pi/2} + \frac{1}{n}\int_0^{\pi/2} \overline{f}'(\theta).\cos(n.\theta).d\theta$$

Le crochet est fait de termes comme $\dfrac{\overline{f(0)}}{n}$ ou $\pm\dfrac{\overline{f(\pi/2)}}{n}$. Ils tendent vers 0 quand n tend vers l'infini.
L'intégrale se majore $\left|\dfrac{1}{n}\int_0^{\pi/2} \overline{f}'(\theta).\cos(n.\theta).d\theta\right| \leqslant \dfrac{1}{n}.Max\{|f'(t)| \mid t \in [0,\pi/2]\}$: on majore le cosinus en valeur absolue par 1 et les $\overline{f}'(\theta)$ par $||\overline{f}'||_\infty$ *($\overline{f}'$ est continue sur un segment, donc majorés)*.
Par encadrement, l'intégrale tend vers 0. La somme I_n tend vers 0 quand n tend vers l'infini.

∘13∘ **Limites en 0^+ et en l'infini de $\left(\dfrac{1+3^x}{2}\right)^{\frac{1}{x}}$.**

Cette application est bien définie au voisinage de 0.
Mais elle y présente une forme indéterminée du type $1^{+\infty}$.
Passons au logarithme : $\dfrac{1}{x}.\ln\left(\dfrac{1+3^x}{2}\right)$. On écrit même $3^x = e^{x.\ln(3)} = 1 + x.\ln(3) + o(x)_{x\to 0}$.

On ajoute 1 et on divise par 2 : $\left(\frac{1+3^x}{2}\right) = 1 + \frac{x.\ln(3)}{2} + o(x)$.
On passe au logarithme $\ln\left(\frac{1+3^x}{2}\right) = \ln\left(1 + \frac{x.\ln(3)}{2} + o(x)\right) = \frac{x.\ln(3)}{2} + o(x)$.
On divise par x : $\frac{1}{x}.\ln\left(\frac{1+3^x}{2}\right) = \frac{1}{x}.\ln\left(1 + \frac{x.\ln(3)}{2} + o(x)\right) = \frac{\ln(3)}{2} + o(1)$.
On passe à la limite qui existe à présent : $\frac{\ln(3)}{2}$.
On passe à l'exponentielle (continue) : $\boxed{\left(\frac{1+3^x}{2}\right)^{\frac{1}{x}} \longrightarrow_{x\to 0} \sqrt{3}}$

La calculatrice confirme cette convergence, l'intuition ne confirmait rien.

En $+\infty$, la forme est en $+\infty^0$, toujours indéterminé.
Le même logarithme donne $\frac{1}{x}.\ln\left(\frac{1+3^x}{2}\right)$ mais cette fois, c'est 3^x qui l'emporte nettement dans le logarithme.
On factorise : $\frac{1}{x}.\Big(\ln(3^x) + \ln\left(\frac{3^{-x}+1}{2}\right)\Big)$.
On simplifie $\ln(3) + \frac{1}{x}.\ln\left(\frac{3^{-x}+1}{2}\right)\Big)$.
Le second terme n'est plus une forme indéterminée, il tend vers 0.
On compose avec l'exponentielle : $\boxed{\left(\frac{1+3^x}{2}\right)^{\frac{1}{x}} \longrightarrow_{x\to+\infty} 3}$

∘14∘

On travaille avec les entiers de 0 à 4 pour les opérations modulo 5. Montrez que $\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$ est nilpotente.
Combien y a-t-il de matrices de taille 2 sur 2 nilpotentes.
Combien y a-t-l de matrices 2 sur 2 de rang 2 ?
Combien y a-t-il de couples de matrices $(A,\ B)$ avec A, B et $A+B$ nilpotentes ?
Le rang est le nombre de colonnes linéairement indépendantes.

$\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}.\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 5 & 10 \\ 10 & 20 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$. pas besoin d'aller chercher loin.

Il y a 5^4 matrices (quatre coefficients à choisir, avec cinq choix pour chacun). C'est le cardinal de notre univers.
Comment y reconnaître les matrices nilpotentes ?
Elles ont un déterminant nul et une trace nulle.

Les matrices de rang 2 sont les matrices inversibles.
Elles sont formées de deux vecteurs colonnes indépendants.
Le premier vecteur colonne ne doit pas être nul, mais c'est la seule contrainte. On remplit comme on veut la première colonne $\begin{pmatrix} a & + \\ c & + \end{pmatrix}$. On a cinq choix pour a et autant pour b. Mais un couple est interdit : $\begin{pmatrix} 0 & + \\ 0 & + \end{pmatrix}$.
On a donc $25-1$ choix du premier vecteur.
Et le second ? Tout $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$...
Sauf $\begin{pmatrix} a & 0 \\ c & 0 \end{pmatrix}, \begin{pmatrix} a & a \\ c & c \end{pmatrix}, \begin{pmatrix} a & 2.a \\ c & 2.c \end{pmatrix}, \begin{pmatrix} a & 3.a \\ c & 3.c \end{pmatrix}, \begin{pmatrix} a & 4.a \\ c & 4.c \end{pmatrix}$. On a cette fois $25-5$ choix.

Et en mettant bout à bout les choix : $(25-1).(25-5)$ matrices inversibles.

∘15∘

Auguste, Baptiste et Clara sortent de colle (d'anglais ou de physique) et indiquent leur note :

Auguste		j'ai 6	j'ai 2 de moins que Baptiste	j'ai 1 de plus que Clara
Baptiste		Clara a 9	je n'ai pas la plus mauvaise note	la différence entre Clara et moi est 3
Clara		Auguste a 7	j'ai moins qu'Auguste	Baptiste a 3 de plus qu'Auguste

Ce n'est pas cohérent tout ça. En effet, chacun a menti une fois et dit la vérité deux fois. Quelle est la note de chacun ?

Les prénoms correspondent à d'anciens élèves. Auguste venait de Lavoisier et termine ses études à SupAero (plus météo et Polytechnico Turin.
Clara venait de Maurice Ravel, a fini l'ENSTA et est développesue fullstack dans le domaine de la santé (comprend qui peut).
Baptiste venait de Marcq-en-Barœul (mais son père habitait à Gambetta !) et est à IMT Atlantique je crois.

On regarde les informations

	1	2	3
Auguste	A=6	A=B-2	A=C+1
Baptiste	C=9	non(Min(A,B,C)=B)	|C-B|=3
Clara	A=7	C<A	B=A+3

On marquera en tout petit sous une barre les assertions fausses. On marquera en grand les affirmations qui sont assurément vraies.
Comme $A1$ et $C1$ sont contradictoire, un des deux est faux (au moins). Ce qui signifie que les autres affirmations de la ligne sont vraies.
• Si Auguste ment en disant "A=6" et Clara sincère en disant "A=7" :

	1	2	3
Auguste	$\overline{A=6}$	A=B-2 donc B=9	A=C+1 donc C=6
Baptiste	C=9	non(Min(A,B,C)=B)	|C-B|=3
Clara	A=7	C<A	$\overline{B=A+3}$

On a trouve $(A,\ B,\ C) = (7,\ 9,\ 6)$. Il faut alors que Baptiste mente une fois sur ses trois affirmations. Il ment en disant C=9 et dit la vérité des ses autres phrases. On peut accepter cette situation.
• Si Auguste est sincère en disant "A=6" et Clara menteuse en disant "A=7" :

	1	2	3
Auguste	A=6	A=B-2	A=C+1
Baptiste	C=9	non(Min(A,B,C)=B)	|C-B|=3
Clara	$\overline{A=7}$	C<A	B=A+3 donc B=9

Comme l'affirmation C2 dit que la plus sale note est C, on déduit que B ment avec B1. Il dit donc vrai avec "B n'est pas le minimum" et avec |C-B|=3 qui force C=6. On a alors $(A,\ B,\ C) = (6,\ 9,\ 6)$. Mais alors A ment deux fois dans ses deux dernière affirmations. On refuse.
• Si Auguste ment avec "A=6" et Clara aussi avec "A=7" ; leurs autres phrases sont sincères :

	1	2	3
Auguste	$\overline{A=6}$	A=B-2	A=C+1
Baptiste	C=9	non(Min(A,B,C)=B)	|C-B|=3
Clara	$\overline{A=7}$	C<A	B=A+3

et la contradiction $A+B-2$ et $B=A+3$ sont en contradiction.
On ne peut donc garder que

Auguste	Baptiste	Clara
7	9	6

∘16∘ ♡ Déterminez $\lim_{x\to 4} \dfrac{x^3-4^3}{x^5-4^5}$.

a et b strictement positifs donnés, déterminez si elle existe la limite de $\dfrac{x^a-\mu^a}{x^b-\mu^b}$ quand x tend vers μ avec $\mu = \dfrac{a+b}{2}$.

On n'écrit pas $\lim_{x\to 4} \dfrac{x^3-4^3}{x^5-4^5}$ *tant qu'on n'a pas prouvé l'existence de la limite. C'est à de tels comportements qu'on détectera chez vous l'élève de Prépas, face à l'élève qui a fait une Terminale destinée aussi à des non-scientifiques (en Terminale, vous étiez avec des élèves se destinant à Médecine, Science-Po et autres... que devait on leur leur enseigner en maths ? A faire des calculs ou à raisonner juste ?* [2]*).*

A faire.

∘17∘ ♡ Comparez les probabilités de "*avoir au moins un* 6 *avec six dés*" et "*avoir au moins deux* 6 *avec douze dés*" (et pourquoi pas "*avoir au moins un* 12 *avec douze dés*" ?).

Comment accéder à la probabilité d'avoir au moins un 6 ? C'est un, ou deux, ou trois... c'est long.

Passons par le complémentaire (idée classique en probabilités).
Le complémentaire, c'est aucun 6. C'est donc que chacun des six dés (indépendants) n'a fait aucun 6 (probabilité $\dfrac{5}{6}$ pour chacun).

$P(aucun\ 6\ avec\ six\ des) = 1-\left(\dfrac{5}{6}\right)^6$ (soit 31031/46656, et environ 0,66 (ou 66pour cent).

---

2. *le programme dit « à faire des calculs », c'est en ça qu'il y a tromperie sur les maths, et sur la formation, mais ça c'est une autre histoire*

De la même façon, le complémentaire de au moins deux 6 avec 12 dés, c'est aucun ou un seul.
On additionne les probabilités de deux événements incompatibles :
$*$ aucun 6 : $\left(\frac{5}{6}\right)^{12}$
$*$ un seul 6 : $12.\left(\frac{1}{6}\right)^{1}.\left(\frac{5}{6}\right)^{11}$ (choix du dé, tirage d'un 6 pour lui, et tirage de « tout sauf 6 » pour les onze autres).
On passe au complémentaire : $1-\left(\frac{5}{6}\right)^{11}-12.\left(\frac{1}{6}\right)^{1}.\left(\frac{5}{6}\right)^{11}$.
Application numérique : 1346704211/2176782336, valeur approchée : 0,62.

Bilan :

au moins un 6 avec six dés	>	au moins deux 6 avec douze dés

Mais de peu !

◦18◦ ♡ Soit f de classe C^3 sur un voisinage de a. Calculez $lim_{h\to 0}\frac{1}{h^2}.\begin{vmatrix} f(a-h) & f(a) \\ f(a) & f(a+h) \end{vmatrix}$ (indication : DL d'ordre 2).

On va utiliser la formule de Taylor Young puisqu'on nous dit que f est C^3 en a :

$$\begin{array}{lllll} f(a+h) & = f(a) & +h.f'(a) & +\frac{h^2.f''(a)}{2} & +\frac{h^3.f^{(3)}(a)}{6} & +o(h^3) \\ f(a-h) & = f(a) & -h.f'(a) & +\frac{h^2.f''(a)}{2} & -\frac{h^3.f^{(3)}(a)}{6} & +o(h^3) \end{array}$$

Le produit $f(a+h).f(a-h)$ contient beaucoup de termes. On se dit que c'est lourd.
On les ramène alors à l'ordre 2, d'autant qu'on va diviser par h^2 :

$$f(a+h).f(a-h)=\left(f(a)+\frac{h^2.f''(a)}{2}+o(h)^2+\frac{h.f'(a)}{1}\right).\left(f(a)+\frac{h^2.f''(a)}{2}+o(h)^2-\frac{h.f'(a)}{1}\right)$$

On développe et on trouve $(f(a))^2+(f(a).f''(a)-(f'(a))^2).h^2+o(h^2)$ en jetant à la poubelle tous les termes inutiles.

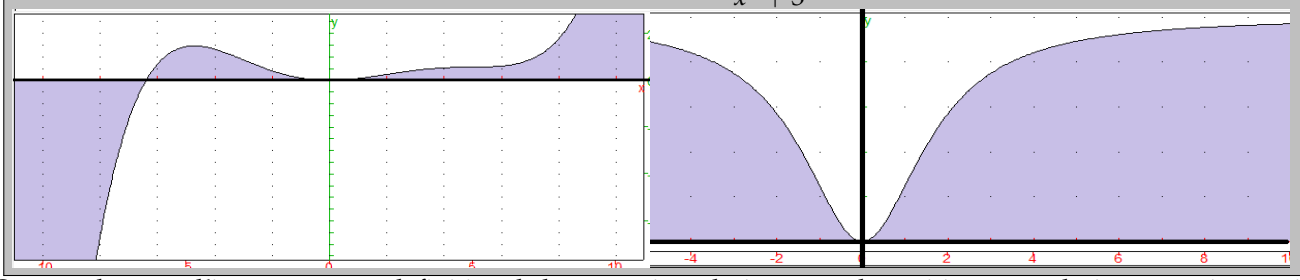
C'est du simple bon sens de ne pas garder des h^3 face à un $o(h^2)$!
C'est la base même des raisonnements que vous devrez mener.
Tout le reste ne sera que calcul. Et donc surtout bon sens.

On soustrait $(f(a))^2$, on divise par h^2 : $lim_{h\to 0}\frac{1}{h^2}.\begin{vmatrix} f(a-h) & f(a) \\ f(a) & f(a+h) \end{vmatrix}=f(a).f''(a)-(f'(a))^2+o(1)_{h\to 0}$.

La limite vaut $\boxed{f(a).f''(a)-(f'(a))^2}$

◦19◦ ♡ Donnez un intervalle le plus grand possible sur lequel $x\longmapsto 3.x^5-20.x^4-110.x^3+900.x^2$ est convexe.

Donnez un intervalle le plus grand possible sur lequel $x\longmapsto\frac{x^2}{x^2+3}$ est convexe.



On prendra pour l'instant comme définition de la convexité « dérive seconde positive » ou « dérivée première croissante ».

On dérive donc deux fois : on factorise 60 :

$$x\longmapsto 60.(x^3-4.x^2-11.x+30)$$

On trouve des racines évidentes : $x\longmapsto 60.(x-5).(x-2).(x+3)$ (calcul).

On dresse un tableau de signes :

x	$]-\infty,\ -3]$	$[-3,\ 2]$	$[2,\ 5]$	$[5,\ +\infty[$
$x-5$	$\ominus$	$\ominus$	$\ominus$	$\oplus$
$x-2$	$\ominus$	$\ominus$	$\oplus$	$\oplus$
$x+3$	$\ominus$	$\oplus$	$\oplus$	$\oplus$
$P''(x)$	$\ominus$	$\oplus$	$\oplus$	$\oplus$

On retient $[5,\ +\infty[$ (et on savait que f'' allait être positive sur un intervalle $[r,\ +\infty[$). Et il reste aussi $[-3,\ 2]$ mais trop petit.
On peut confirmer sur le dessin, entre les points d'inflexion.

Du calcul encore : $\left(x\longmapsto \dfrac{x^2}{x^2+3}\right)'' = \left(x\longmapsto x^2.(x^2+3)^{-1}\right)''$

$$\left(x\longmapsto \frac{x^2}{x^2+3}\right)'' = \left(x\longmapsto 2.(x^2+3)^{-1}+2.2.x.(-2.x.(x^2+3)^{-2})+x^2.(-2.(x^2+3)^{-2}+8.x^2.(x^2+3)^{-3})\right)$$

(danke Leibniz)
Le dénominateur est positif, il reste un numérateur en

$$2.(x^2+3)^2-8.x^2.(x^2+3)+x^2.(-2.(x^2+3)+8.x^2)$$

Trop fort : $18-18.x^2$!
La réponse est donc $[-1,\ 1]$.

◦20◦

♣☺ On définit : $I=\displaystyle\int_0^{\pi/4}\ln(1+\tan(t)).dt$, $J=\displaystyle\int_0^{\pi/4}\ln(\cos(x)).dx$, $K=\displaystyle\int_0^{\pi/4}\ln(\cos(x-\pi/4)).dx$ et $L=\displaystyle\int_0^{\pi/4}\ln(\sin(x)+\cos(x)).dx$.
Montrez : $I=L-J$, $J=K$. Calculez $L-K$ par trigonométrie. Déduisez la valeur de I.

Toutes les fonctions sous le signe somme sont continues. Toutes les intégrales existent.
On calcule $L-J$: $\displaystyle\int_0^{\pi/4}\ln(\sin(x)+\cos(x)).dx-\int_0^{\pi/4}\ln(\cos(x)).dx=\int_0^{\pi/4}\Big(\ln(\sin(x)+\cos(x))-\ln(\cos(x))\Big).dx$ par linéarité.

Et justement, $\ln(\cos(x)+\sin(x))-\ln(\cos(x))=\ln\left(\dfrac{\cos(x)+\sin(x)}{\cos(x)}\right)=\ln\Big(1+\tan(x)\Big)$. On a bien I.

On part de $J=\displaystyle\int_0^{\pi/4}\ln(\cos(x)).dx$ et on veut trouver $K=\displaystyle\int_0^{\pi/4}\ln(\cos(x-\pi/4)).dx$. Sachant que le développement de $\cos(x-\pi/4)$ donne autre chose, on se demande si on n'a pas un changement de variable à faire.

On part de J et on pose $\theta=\dfrac{\pi}{4}-x$. On a alors $J=\displaystyle\int_{x=0}^{x=\pi/4}\ldots=\int_{\theta=\pi/4}^{\theta=0}\ldots=-\int_{\theta=0}^{\theta=\pi/4}\ldots$. On a aussi $d\theta=-dx$, ce qui va rétablir les bornes.

On a donc $J=\displaystyle\int_0^{\pi/4}\ln(\cos(x)).dx=-\int_{\theta=\pi/4}^{\theta=0}\cos(\pi/4-\theta).d\theta$. La seconde forme est bien celle de $K=\displaystyle\int_0^{\pi/4}\ln(\cos(x-\pi/4)).dx$, avec une variable qui ne s'appelle plus x mais θ. Mais les variables sont muettes.

On regarde $K=\displaystyle\int_0^{\pi/4}\ln(\cos(x-\pi/4)).dx$. On développe par formule de trigonométrie :

$$K=\int_0^{\pi/4}\ln\left(\frac{\sqrt{2}}{2}.\cos(x)+\frac{\sqrt{2}}{2}.\sin(x)\right).dx$$

On sépare par propriété du logarithme : $K=\displaystyle\int_0^{\pi/4}\Big(\ln(\cos(x)+\sin(x))+\ln(\sqrt{2}/2)\Big).dx$.

On sépare et on intègre $\displaystyle\int_0^{\pi/4}\ln(\sqrt{2}/2).dx=\frac{\pi}{4}.\ln(\sqrt{2}/2)$.

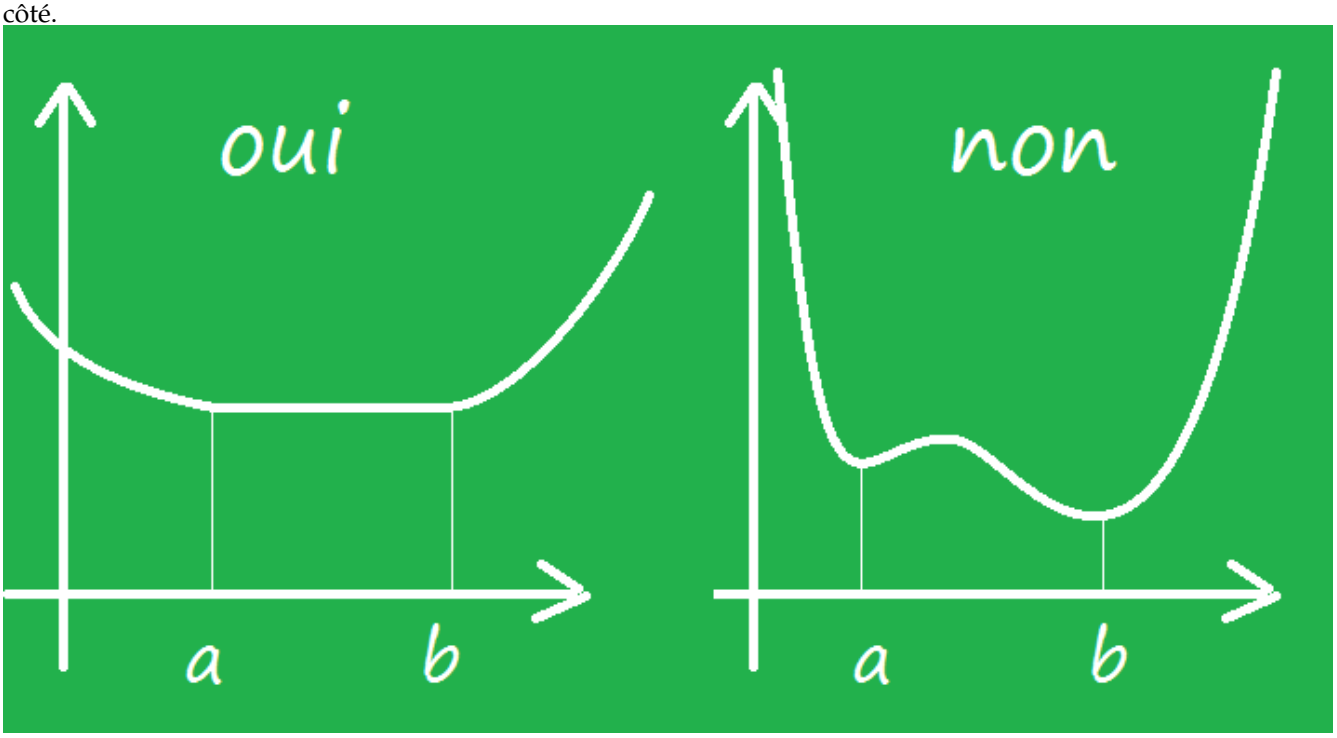
Finalement, $\boxed{K=L-\dfrac{\pi.\ln(2)}{8}}$

On rassemble tout ce qu'on sait : $I=L-J$, $J=K$ et $K=L-\dfrac{\pi.\ln(2)}{8}$.

On assemble : $\boxed{I=\dfrac{\pi.\ln(2)}{8}}$ et tout ça sans primitive !

∘21∘ On suppose que f convexe de $\mathbb{R}$ dans $\mathbb{R}$ admet un minimum local en a et aussi en b. Montrez alors $f(a) = f(b)$ et montrez que f est constante sur $[a,\ b]$.

On va montrer que f est de la forme suivante et ne peut pas pour cause de convexité être des formes indiquées à côté.



Première version : f est dérivable (simpliste mais c'est déjà ça).
Comme f admet un minimum local en a, sa dérivée est nulle en a.
Comme f admet un minimum local en f, sa dérivée s'annule en b.
Comme f est convexe, sa dérivée est croissante.
Ayant $f'(a) = f'(b) = 0$ et f' croissante, pour tout x entre a et b, on a $0 = f'(a) = f'(x) = f'(b) = 0$.
Comme f' est nulle sur tout l'intervalle $[a,\ b]$, $f$est constante sur $[a,\ b]$ (et ceci nous donne $f(a) = f(b)$ qu'on n'avait pas forcément dès le départ).

De plus, le graphe est au dessus de ses tangentes. Or, la tangente en $\dfrac{a+b}{2}$ est horizontale. La fonction est donc toujours plus grande que ce minimum qui passe du statut de minimum local à celui de minimum global.

Preuve en n'utilisant que la définition « brute » et aucune hypothèse supplémentaire de dérivabilité.
On a quand même des dérivabilités latérales.
Si l'on a un minimum en a, f est dérivable à droite et à gauche en a car convexe.
Comme on a un minimum en a, les taux d'accroissement ont un signe connu à droite (localement) et à gauche.
On obtient donc $f'_d(a) \geqslant 0$.
De même, à gauche de b, on a $\dfrac{f(x)-f(b)}{x-b} \leqslant 0$ à cause du minimum, puis en passant à la limite monotone : $f'_g(b) \leqslant 0$.

Mais on a aussi

$$f'_d(a) \leqslant \frac{f(b)-f(a)}{b-a} \leqslant f'_g(b)$$

par convexité.
En tenant compte des signes de chacun, la seule possibilité est « tous nuls ».
On en déduit déjà $f'_d(a) = f'_g(b) = 0$ mais aussi $f(a) = f(b)$. C'est le même minimum local pour les deux.

Mais de plus, pour tout x entre a et b, on a $f_d(a) \leqslant f'_g(x) \leqslant f'_d(x) \leqslant f'_g(b)$ (c'est dans le cours, croissance de la dérivée).
On en déduit $f'_g(x) = f'_d(x) = 0$ et f est constante sur l'intervalle $[a,\ b]$ (dérivée identiquement nulle).

Vous cernez dans cet exercice la distinction entre « la dérivée s'annule (en un point) » et « la dérivée est nulle (sur un intervalle) ».

Peut être l'avez vous eue par les taux d'accroissements uniquement.

∘22∘

Montrez que $x \longmapsto \ln(1+e^x)$ est convexe.
Déduisez pour a et b positifs : $1+\sqrt{a.b} \leqslant \sqrt{(1+a).(1+b)}$ et plus généralement
Déduisez : $1+\left(\prod_{k=1}^{n} x_k\right)^{\frac{1}{n}} \leqslant \left(\prod_{k=1}^{n}(1+x_k)\right)^{\frac{1}{n}}$ pour des x_k strictement positifs.
Déduisez $\sqrt[n]{a_1\dots a_n}+\sqrt[n]{b_1\dots b_n} \leqslant \sqrt[n]{(a_1+b_1)\dots(a_n+b_n)}$ pour des réels strictement positifs a_j et b_i.
$1+\sqrt[n]{a_1\dots a_n} \leqslant \sqrt[n]{(1+a_1)\dots(1+a_n)}$.

Il suffit de dériver une fois : $x \longmapsto \dfrac{e^x}{e^x+1}$ est la composée de deux applications croissantes : $x \longmapsto e^x$ et $t \longmapsto \dfrac{t}{t+1}$.
Si vous y tenez, dérivez deux fois, mais je serai déçu.

On sent la petit inégalité de convexité à deux termes dans le cas particulier $t=\dfrac{1}{2}$: $f\left(\dfrac{a+b}{2}\right) \leqslant \dfrac{f(a)+f(b)}{2}$.
On a donc

$$\ln\left(1+e^{\frac{x+y}{2}}\right) \leqslant \frac{\ln(1+e^x)+\ln(1+e^y)}{2} = \ln\left(\sqrt{(1+e^x).(1+e^y)}\right.$$

Si l'on compose avec le logarithme (croissant) :

$$1+\sqrt{e^x.e^y} \leqslant \sqrt{(1+e^x).(1+e^y)}$$

0n a compris. On l'applique à $x=\ln(a)$ et $\ln(b)$ puisque a et b sont donnés strictement positifs.

$$1+\sqrt{a.b} \leqslant \sqrt{(1+a).(1+b)}$$

On note que si on avait essayé de mettre bout à bout des inégalités de convexité, on échouait car exp *est convexe mais* ln*concave.*

On le rédige mieux pour n points.
Les données sont les x_k, c'est donc d'eux qu'on part.
On pose alors $y_k=\ln(x_k)$. On a alors n réels.
On écrit pour eux la grande inégalité de convexité pour un isobarycentre[3] :

$$f\left(\frac{y_1+\ldots+y_n}{n}\right) \leqslant \frac{f(y_1)+\ldots f(y_n)}{n} = \frac{\ln(1+x_1)+\ldots+\ln(1+x_n)}{n} = \ln\left(\sqrt[n]{(1+x_1)\dots(1+x_n)}\right)$$

On passe à l'exponentielle *(croissante)* : $1+e^{\frac{y_1+\ldots+y_n}{n}} \leqslant \sqrt[n]{(1+x_1)\dots(1+x_n)}$
On simplifie le premier membre et on a la formule.

Une question sans grande difficulté. Mais qui fera le tri le jour du concours.
Si vous vous contentez de formules et oubliez oubliez de préciser que vous composez par l'exponentielle en précisant qu'elle est croissante, vous allez perdre un tiers des points.
Si vous partez des $y_i$au lieu de partir des x_i, vous perdez un sixième des points (qui est donné ?).

Une fois donnés les a_i et les $b_j$strictement positifs, il suffit ensuite de voir que l'inégalité

$$\sqrt[n]{a_1\dots a_n}+\sqrt[n]{b_1\dots b_n} \leqslant \sqrt[n]{(a_1+b_1)\dots(a_n+b_n)}$$

est équivalente à

$$1+\sqrt[n]{\frac{b_1}{a_1}.\frac{b_2}{a_2}\dots\frac{b_n}{a_n}} \leqslant \sqrt[n]{\frac{(a_1+b_1)}{a_1}\dots\frac{(a_n+b_n)}{a_n}}$$

en divisant tout par $\sqrt[n]{a_1\dots a_n}$.
Il suffit donc d'appliquer le résultat précédent aux $\dfrac{b_k}{a_k}$.

---

3. « isobarycentre » est le grand mot pour dire « moyenne avec tous les coefficients égaux » ou même « moyenne » quand on est prof d'histoire géo et qu'on a du mal avec ces histoires de coefficients

◦23◦

Montrez que si f est convexe, alors pour tout a positif, $t \longmapsto e^{a.f(t)}$ est convexe.

Réciproquement, on suppose que pour tout a positif, $t \longmapsto e^{a.f(t)}$ est convexe. Montrez alors que $t \longmapsto \dfrac{e^{a.f(t)}-1}{a}$ est convexe. Déduisez que f est convexe.

Pour le sens classique, c'est de la composition « convexe avec convexe croissante ».
On travaille avec la petite inégalité de convexité.
Pour x et y donnés, et pour t entre 0 et 1 : $f((1-t).x+t.y) \leqslant (1-t).f(x)+t.f(y)$.
On multiplie par a *(positif)* et on passe à l'exponentielle *(croissante)* : $e^{f((1-t).x+t.y)} \leqslant e^{(1-t).a.f(x)+t.a.f(y)}$.
On exploite la convexité de l'exponentielle :

$$e^{f((1-t).x+t.y)} \leqslant e^{(1-t).a.f(x)+t.a.f(y)} \leqslant (1-t).e^{a.f(x)}+t.e^{a.f(y)}$$

On a obtenu la convexité de $x \longmapsto e^{a.f(x)}$.

Ce que guettera le correcteur aux concours, c'est ce qui est dans les parenthèses : croissante, positif, convexe.

La réciproque demande plus de prudence.
C'est le « pour tout a qui va nous servir ». rappelons en effet que $x \longmapsto \ln(x)$ n'est pas convexe alors que $x \longmapsto e^{\ln(x)}$ est convexe (mais pas chaque $x \longmapsto e^{a.\ln(x)}$).
On ne peut pas dériver des inégalités, ça ne nous apporte rien.

Alors que faire ? Supposons que pour tout a, $e^{a.f}$ soit convexe.

On fait une transformation affine, qui ne modifie pas la convexité : $x \longmapsto \dfrac{e^{a.f(x)}-1}{a}$ est convexe pour tout a.
On passe à la limite quand a tend vers 0 : $x \longmapsto f(x)$ est convexe.
Toc toc bada boum, c'est fini.

Un détail qui manque : $\dfrac{e^{a.f(x)}-1}{a}$ *tend vers* $f(x)$ *quand* a *tend vers* 0.
Oui, il suffit d'écrire un développement limité : $e^{a.f(x))} = 1 + a.f(x) + o(a.f(x))_{a\to 0}$ *donc* $\dfrac{e^{a.f(x))}-1}{a} = f(x)+o(1)_{a\to 0}$.
On pouvait aussi l'avoir en retrouvant un taux d'accroissement $\dfrac{e^{a.f(x)}-1}{a.f(x)}.f(x)$.

L'autre détail : *une limite de fonctions convexes est convexe.*
On se donne x, y *et* t (t *est entre* 0 *et* 1).
On écrit pour tout n : $f_n((1-t).x+t.y) \leqslant (1-t).f_n(x)+t.f_n(y)$
On fait tendre n *vers l'infini, les inégalités strictes deviennent larges, les inégalités larges le restent :* $f((1-t).x+t.y) \leqslant (1-t).f(x)+t.f(y)$

◦24◦

f est de classe C^2 et convexe. Montrez que $x \longmapsto \displaystyle\int_{t=x-1}^{x+1} f(t).dt$ est aussi convexe.

Une hypothèse nous est soufflée qui nous met sur une piste simple : « f est C^2 ».
Le critère est donc double : f' est croissante
f'' est positive
et on n'a pas besoin d'écrire les inégalités de trois cordes.
L'application $x \longmapsto \displaystyle\int_{t=x-1}^{x+1} f(t).dt$ est à son tour dérivable par continuité des f.

$$\left(x \longmapsto \int_{t=x-1}^{x+1} f(t).dt\right)' = \Big(x \longmapsto f(x+1)-f(x-1)\Big)$$

$$\left(x \longmapsto \int_{t=x-1}^{x+1} f(t).dt\right)'' = \Big(x \longmapsto f'(x+1)-f'(x-1)\Big)$$

Ça y est, vous y voyez clair ? f' est croissante, donc $\forall x,\ f'(x+1)-f'(x-1) \geqslant 0$.
La dérivée seconde est positive, l'application $\left(x \longmapsto \displaystyle\int_{t=x-1}^{x+1} f(t).dt\right)$ est convexe.

Autre approche : une intégrale est une somme « infinie ». Et en écrivant $\left(x \longmapsto \int_{u=-1}^{1} f(x+u).dt\right)$ *on a une somme d'applications convexes* $x \longmapsto f(x+u)$.

◦25◦ On définit $f = t \longmapsto (t^2+1).e^{-t}$. Donnez l'intervalle le plus grand possible sur lequel f est croissante et convexe.
Même question avec décroissante et convexe.
Même question avec décroissante ou convexe.

Il s'agit d'étudier le signe de f' (monotonie) et de f'' (convexité).
On dérive donc, et plutôt deux fois qu'une.

	valeur	du signe de	positive sur
$f(x)$	$(1+x^2).e^{-x}$	$1+x^2$	$\mathbb{R}$
$f'(x)$	$-(x^2-1)^2.e^{-x}$	$(-x^2-1)$	rien
$f''(x)$	$(x^2-4.x+3).e^{-x}$	$(x-1).(x-3)$	$]-\infty,\ 1]$ et $[3,\ +\infty[$
décroissante et convexe			$]-\infty,\ 1]$ et $[3,\ +\infty[$
décroissante ou convexe			partout

Attention, on ne dit pas convexe sur $]-\infty,\ 2] \cup [3,\ +\infty[$.
Ca n'a pas de sens. la convexité (la croissance aussi) ne se définit que sur un intervalle à la fois.
Prenez cette habitude...

◦26◦ ♡ Sur quel(s) intervalle(s) $t \longmapsto (1-t^2).e^t$ est elle convexe ?

Il suffit d'étudier le signe de sa dérivée seconde : $t \longmapsto (1-t^2).e^t$
$t \longmapsto (1-t^2-2.t).e^t$
$t \longmapsto (-t^2-4.t-1).e^t$

On demande donc $t^2+4.t+1 \leqslant 0$.
On trouve $\boxed{[2-\sqrt{3},\ 2+\sqrt{3}]}$

Alors pourquoi ce pluriel ? Il n'y a qu'un intervalle, non ?

Non. Par exemple $[-2,\ -1]$ convient aussi. Puisque $[-2,\ -1] \subset [2-\sqrt{3},\ 2+\sqrt{3}]$.
En fait, la réponse est « tout intervalle inclus dans $[2-\sqrt{3},\ 2+\sqrt{3}]$.

◦27◦ ♡ Calculez ces sommes multiples pour n donné $\boxed{A_n = \sum_{0\leqslant j<i\leqslant n} \frac{j}{i}}$ $\boxed{B_n = \sum_{i+j=n} i.j}$ $\boxed{C_n = \sum_{\substack{0\leqslant i\leqslant n \\ 0\leqslant j\leqslant n}} Max(i,\ j)}$
Si vous n'avez pas confiance, écrivez un script Python qui prend en entrée n et les calcule.

$$A_n = \sum_{0\leqslant j<i\leqslant n} \frac{j}{i}$$
$$A_n = \sum_{i=1}^{n} \frac{1}{i}.\sum_{j=0}^{i-1} j$$
$$A_n = \sum_{i=1}^{n} \frac{1}{i}.\frac{i.(i-1)}{2}$$
$$A_n = \sum_{i=1}^{n} \frac{i-1}{2}$$
$$\boxed{A_n = \frac{n.(n-1)}{4}}$$

$$B_n = \sum_{i+j=n} i.j$$
$$B_n = \sum_{i=0}^{n} i.(n-i)$$
$$B_n = n.\sum_{i=0}^{n} i - \sum_{i=0}^{n} i^2$$
$$\boxed{B_n = \frac{n^3-n}{6}}$$

$C_n = \sum_{\substack{0\leqslant i\leqslant n\\ 0\leqslant j\leqslant n}} Max(i,\ j)$
$C_n = \sum_{0\leqslant i<j\leqslant n} Max(i,\ j) + \sum_{0\leqslant i=j\leqslant n} Max(i,\ j) + \sum_{0\leqslant j<i\leqslant n} Max(i,\ j)$
$C_n = 2.\sum_{0\leqslant i<j\leqslant n} Max(i,\ j) + \sum_{0\leqslant i=j\leqslant n} i$
$C_n = 2.\sum_{0\leqslant i<j\leqslant n} j + \sum_{i=0}^{n} i$
$C_n = 2.\sum_{j=0}^{n} j\sum_{i=0}^{j-1} 1 + \frac{n.(n+1)}{2}$
$C_n = 2.\sum_{j=0}^{n} j^2 + \frac{n.(n+1)}{2}$
$$\boxed{C_n = \frac{n.(n+1).(4.n+5)}{6}}$$

∘28∘ ♡ Montrez pour tout x réel : $e^x \geqslant 1 + x + \frac{x^2}{2} + \frac{x^3}{6}$.

x donné, on écrit la formule de Taylor avec reste intégrale pour l'exponentielle à l'ordre 3 entre 0 et x :

$$e^{0+x} = \sum_{k=0}^{3} \frac{\exp^{(k)}(0)}{k!}.x^k + \frac{x^4}{3}.\int_0^1 (1-t)^3.\exp^{(4)}(t.x).dt = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{3}.\int_0^1 (1-t)^3.e^{t.x}.dt$$

Le reste intégrale est positif (exposant pair). On a la minoration.

C'est le même modèle que $e^x \geqslant 1+x$.
On aurait pû aussi, masochistement, définit $f = x \longmapsto e^x - 1 - x - \frac{x^2}{2} - \frac{x^3}{6}$.
L'application $f^{(4)}$ est l'exponentielle, ositive.
$f^{(3)}$ est donc croissante, et elle est nulle en 0 ($f^{(3)} = x \longmapsto e^x - 1$).
$f^{(3)}$ est donc d'abord négative, puis positive.
Il s'ensuite que f'' est décroissante puis croissante.
f'' admet un minimum en 0. Et il vaut 0.
f'' est donc positive.
f' est donc croissante. Comme elle s'annulle en 0, elle est donc négative puis positive.
f est donc décroissante puis croissante.
f admet en 0 un minimum, égal à 0 (simple calcul).
f est donc positive. C'est ce qu'on voulait.

∘29∘ Soit f deux fois dérivable avec f'' positive. Montrez que si $f'(a)$ est nul, alors f admet un minimum en a.

f' est donc croissante.
Comme elle est nulle en a, elle est donc négative (ou nulle) avant, puis positive après.
f est doonc décroissante, puis croissante.
Bref, c'est un tableau de variations qui nous dit que f admet un minimum en a.

Rappelons qu'avec l'annulation de f' ont n'a pas forcément un minimum.
Oui, je sais, ça peut être un maximum.
Mais ce peut être aussi ni l'un ni l'autre, comme avec $x \longmapsto x^3$ e 0.
Le critère est « annulation et changement de signe de la dérivée ».

Mais ici, on peut aussi écrire $f(x) = f(a) + 0.(x-a) + \frac{(x-a)^2}{1}\int_0^1 (1-t).f''(t.x + (1-t).a).dt$.
l'intégrale est positive par positivité de f''. Le carré devant l'intégrale l'est aussi.
La différence $f(x) - f(a)$ est donc positive.
On a bien un minimul en a.

∘30∘ Source : d'après un oral de Centrale. On considère $n \geqslant 2$ joueurs, numérotés de 1 à n, participant à un tournoi où chacun affronte tous les autres, sans égalité possible dans une rencontre. On définit la matrice $A = (a_i^k)_{i\leqslant n, k\leqslant n}$ de la manière suivante : $a_i^i = 0$, $a_i^k = 1$ si i a gagné contre k et $a_i^k = -1$ si k a gagné contre i. Ecrivez une procédure renvoyant pour n donné une matrice de tournoi aléatoire.

On doit créer des matrices telles que $\begin{pmatrix} 0 & -1- & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 & 1 \\ 1 & -1 & 0 & 1 & 1 \\ -1 & 1 & -1 & 0 & -1 \\ 1 & -1 & -1 & 1 & 0 \end{pmatrix}$. Comment les remplir ?

On a deux démarches. Comme dans le cours, je vais préférer la méthode dynamique où l'on fabrique les lignes une à une.

```
def Alea(n) :
....M = [] #initialisation vide
....for i in range(n) : #ligne par linge
........L = [] #on commence avec une ligne vide
........for k in range(n) : #on avance case à case
............hum hum #là, il y a du travail
........M += L #la ligne est finie, on la colle
....return M #la matrice est finie, on la retourne
```

Il reste à voir comment on complète les lignes, sachant que la matrice doit être antisymétique. On ne va donc créer que les termes au dessus de la diagonale ; ceux qui sont au dessous sont obtenus par passage à l'opposé.
Quand `k` est plus grand que `i` *(fin de ligne)*, on tire un nombre au hasard *(quitte à avoir créé déjà une fonction qui choisit `1` ou `-1`)*.
Quand `k` est égal à `i`, on met `0`.
Quand `k` est plus petit que `i` *(début de ligne)*, on recopie au signe près un terme déjà créé : `-M[k][i]`.
Donc déjà, le créateur de `-1` ou `1` :

```
def pm() : #comme plus ou moins 1
....return 2*randrange(2)-1
```

Pui la matrice elle même, suivant le schéma indiqué :

```
def Alea(n) :
....M = []
....for i in range(n) :
........L = []
........for k in range(n) :
............if k < i :
................L.append(-M[k][i])
............if k == i :
................L.append(0)
............if k > i :
................L.append(pm())
........M.append(L)
....return M
```

On peut aussi effectuer une création statique. On crée d'abord une matrice avec des `0`. On balaye ensuite la moitié supérieure, on y met des `1` ou `-1`, et on complète dans le même temps la moitié inférieure

```
def Alea(n) :
....M = [[0 for k in range(n)] for i in range(n)]
....for i in range(n) :
........for k in range(i-1) :
............M[i][k] = pm()
............M[k][i] = -pm()
....return M
```

Evidemment, la ligne `M[k][i] = -pm()` est une erreur. Elle relance un nouveau tirage aléatoire. Il faut reprendre le précédent : `a = pm()`
`M[i][k], M[k][i] =a,`
`-a`

Voici une procédure *(lourde, de complexité n!)* de calcul de déterminant, mais il manque une sous-procédure ; écrivez la :

```
def det(M) :
....if len(M)==0 :
........return 1
....S, signe = 0, 1
....for i in range(len(M)) :
........S += signe*M[i][0]*det(delRowCol(M,i,0))
....signe = -signe
....return S
```

Que fait la procédure ?

Elle prend en entrée une matrice. Si la matrice est de taille nulle, son déterminant vaut 1, le programme retourne 1 *(et n'exécute pas la suite)*.
Si la matrice est de taille positive, on passe à la suite.

On initialise une somme `S` à `0` *(on l'incrémentera par* `S+=...`*)*, et un clignotant à `1` *(on le fera clignoter en* `sigme=-signe`*)*.

On parcourt la première colonne de la matrice avec des `M[i][0]`.

Chacun est multiplié par le déterminant d'une matrice à laquelle il manque la ligne `i` et la colonne `0`, ce sont les cofacteurs.

On reconstruit donc la formule

$$\det(M) = \sum_{k=0}^{n-1} (-1)^k . m_i^k . cof_i^k$$

L'énoncé indexait ses matrices de 1 *à* n *tandis que Python le fait de* 0 *à* $n-1$*. Mais comme tout est visuel ou finalement calculatoire sur les déterminants, je ne pense pas que ce soit important qu'on décalle les indices.*

Il manque donc la procédure `delRowCol` [4] qui prend trois arguments : une matrice `M` et deux indices. On doit effacer la ligne d'indice `i` et la colonne d'indice `k`.
On peut relire les lignes une à une avec un indice `ii` qui avance *(on sautera le cas ii==i)* et pour chaque ligne, on lira les termes un par un grâce à un indice `kk` *(qui évitera la valeur* `k`*)*.

```
def delRowCol(M, i, k) :
....MM = []
....for ii in range(len(M)) :
........LL = []
........if ii != i :
............for kk in range(len(M[ii])) :
................if kk != k :
....................LL.append(M[ii][kk])
............MM.append(LL)
....return MM
```

On peut aussi utiliser la méthode `pop` qui enlève un terme sur une liste (`L.pop(k)` efface l'élément d'indice `k` de la liste `L`).

```
def delRowCol(M,i,k) :
....MM = [L[:] for L in M] #crée une copie de M
....M.pop(i) #on efface une ligne
....for L in MM : #on prend les lignes une à une
........L.pop(k) #on pop chaque lignes
....return MM
```

Efficace, non ?

On constate pour n impair que ces déterminants sont toujours nuls. Prouvez le.

Ces matrices de tournois sont antisymétries par construction. Elles vérifient ${}^tM = -M$. On passe au déterminant : $\det({}^tM) = \det(-M)$. On simplifie avec ce qu'on sait : $\det(M) = \det({}^tM) = (-1)^n . \det(M)$.
Pour n impair, on obtient $\det(M) = -\det(M)$ d'où $\det(M) = 0$.

On retient : les matrices antisymétriques de format impair sont non inversibles (déterminant nul). Les mauvais élèves ignorent ce résultat et sont ébahis qu'on leur demande de le prouver. Les mauvais élèves oublient que ce n'est vrai qu'en format impair. Les bons élèves voient/retiennent la preuve et tiennent à la fois formule, cadre d'application et preuve...

$J_n = (1)_{i\leqslant n, k\leqslant n}$. Calculez $\det(J_n - I_n)$.

On écrit la forme de la matrice J_n : que des 1, puis de la matrice $J_n - I_n$: que des 1 sauf la diagonale nulle. On rédige la preuve avec la matrice de taille 5 :

4. delete Row (=ligne) Column (=colonne)

$$\begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{vmatrix}$$ on soustrait la première ligne à chacune des autres

$$\begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{vmatrix} = \begin{vmatrix} 4 & 1 & 1 & 1 & 1 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{vmatrix}$$ on somme toutes les colonnes sur la première

La matrice obtenue est triangulaire, son déterminant est le produit des termes diagonaux : $\boxed{(n-1).(-1)^{n-1}}$

Soient M et N deux matrices coefficients entiers telles que $M - N$ ait tous ses coefficients pairs. Montrez que $\det(M)$ et $\det(N)$ sont deux entiers de même parité.

On prend deux matrices à coefficients entiers de termes généraux m_i^k et n_i^k. On suppose que pour tout couple $(i,\ k)$, m_i^k et n_i^k sont de même parité *(différence paire)*.

Pour tout i et tout σ on a $m_i^{\sigma(i)} = n_i^{\sigma(i)}\ mod\ 2$. On multiplie $\prod_{i=1}^{n} m_i^{\sigma(i)} = \left(\prod_{i=1}^{n} n_i^{\sigma(i)}\right)\ mod\ 2$ *(compatibilité des congruences et de la multiplication)*. On multiplie par la signature et on somme : $\left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} m_i^{\sigma(i)}\right) = \left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} n_i^{\sigma(i)}\right)\ mod\ 2$. Les deux déterminants ont la même parité.

Façon PSI : $\left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} m_i^{\sigma(i)}\right) = \left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} (n_i^{\sigma(i)} + 2.k_i^{\sigma(i)})\right)$ et on développe.

Façon PC : $\left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} m_i^{\sigma(i)}\right) = \left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} (n_i^{\sigma(i)} + 2.k)\right)$ et on développe (mais on n'a pas compris que ce k dépendait de la position, confus comme vous dans les variables).

Façon MP : dans la formule $\left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} m_i^{\sigma(i)}\right)$, il n'y a que des sommes et produits, elles sont compatibles avec les congruences modulo 2.

A vous de choisir la rédaction que vous préférez et estimez la plus claire.

Déduisez que les matrices de tournois aléatoires en taille paire sont inversibles.

On prend une matrice de tournoi aléatoire M de taille n. Les deux matrices $J_n - I_n$ et M sont à coefficients entiers, et ont terme à terme la même parité *(1 ou -1 hors de la diagonale, et 0 sur la diagonale)*.
Les deux matrices ont le même déterminant modulo 2.
On a donc $\det(M) = (n-1).(-1)^{n-1}\ mod\ 2$.
Si n est pair, $(n-1).(-1)^{n-1}$ est impair.
$\det(M)$ est donc impair. Il ne peut pas être nul. La matrice M est inversible.

∘31∘

Petits exercices du cours :
a- La réciproque d'une application convexe est elle convexe ? concave ?
Qu'allez vous utiliser ? Déterminant ? Trois cordes ? Petite inégalité ? Grande inégalité ?
Et d'ailleurs, la réciproque existe-t-elle ?

Ce qui passe : •si f est convexe et strictement croissante alors f^{-1} est concave.
(pensez à $x \longmapsto e^x$ et sa réciproque le logarithme).
•si f est convexe et strictement croissante alors f^{-1} est concave.
(pensez à $x \longmapsto e^x$ et sa réciproque le logarithme).
• si f est à la fois convexe et concave (affine), sa réciproque est affine (convexe et concave).

Méthode : trois cordes.
rapide : $\begin{vmatrix} 1 & 1 & 1 \\ a & b & c \\ f(a) & f(b) & f(c) \end{vmatrix} = -\begin{vmatrix} 1 & 1 & 1 \\ f(a) & f(b) & f(c) \\ a & b & c \end{vmatrix} = -\begin{vmatrix} 1 & 1 & 1 \\ f(a) & f(b) & f(c) \\ f^{-1}(f(a)) & f^{-1}(f(b)) & f^{-1}(f(c)) \end{vmatrix}$ pour f croissante.

Et pour f décroissante, il faut intervertir les deux colonnes.

b- Si f est convexe et g convexe et croissante, alors $g \circ f$ est convexe. Pour démontrer ça, qu'allez vous utiliser ? déterminant ? Trois cordes ? Petit inégalité ? grande inégalité ?

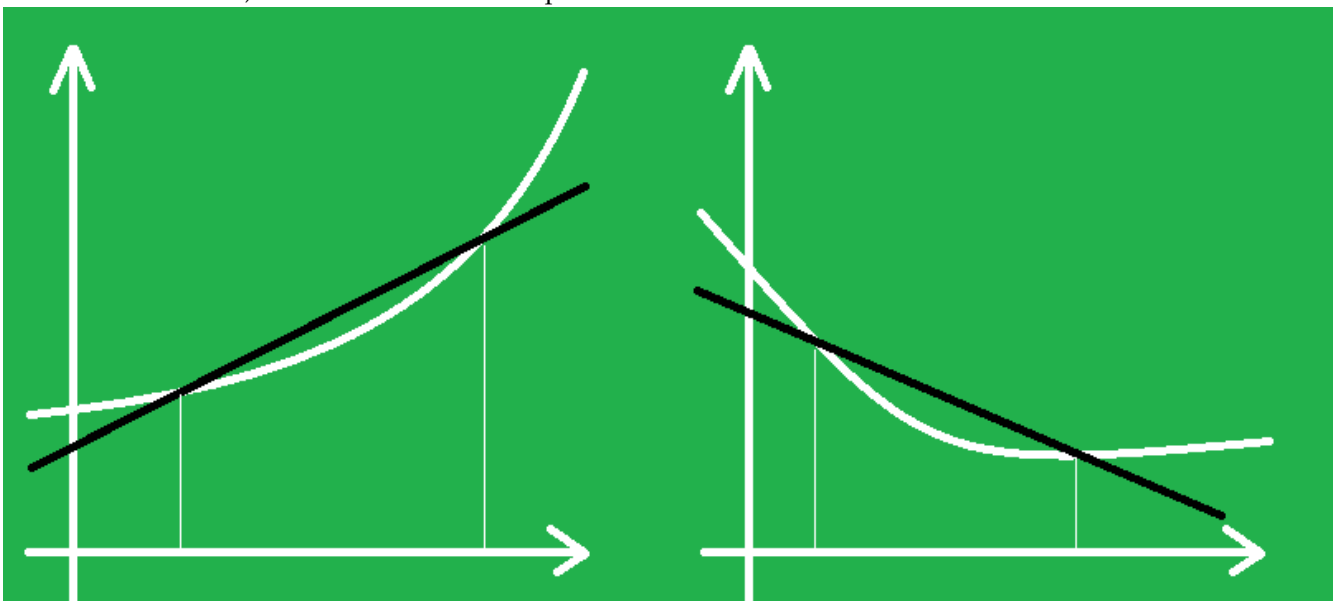
Petite inégalité.
On se donne x et y puis t entre 0 et 1 et on écrit successivement

$$f((1-t).x+t.y) \leqslant (1-t).f(x)+t.f(y) \qquad f \text{ convexe}$$
$$g\big(f((1-t).x+t.y)\big) \leqslant g\big((1-t).f(x)+t.f(y)\big) \qquad g \text{ croissante}$$
$$g\big(f((1-t).x+t.y)\big) \leqslant g\big((1-t).f(x)+t.f(y)\big) \leqslant (1-t).g\big(f(x)\big)+t.g\big(f(y)\big) \qquad g \text{ convexe}$$

Combien d'élèves tombent dans le piège et se contentent d'affirmer « f et g convexes donc $g \circ f$ convexe ».
Alors qu'il suffirait de les convaincre par f convexe et $g = x \longmapsto -x$...

c- Une application convexe de $\mathbb{R}$ dans $\mathbb{R}$ peut elle être majorée ?

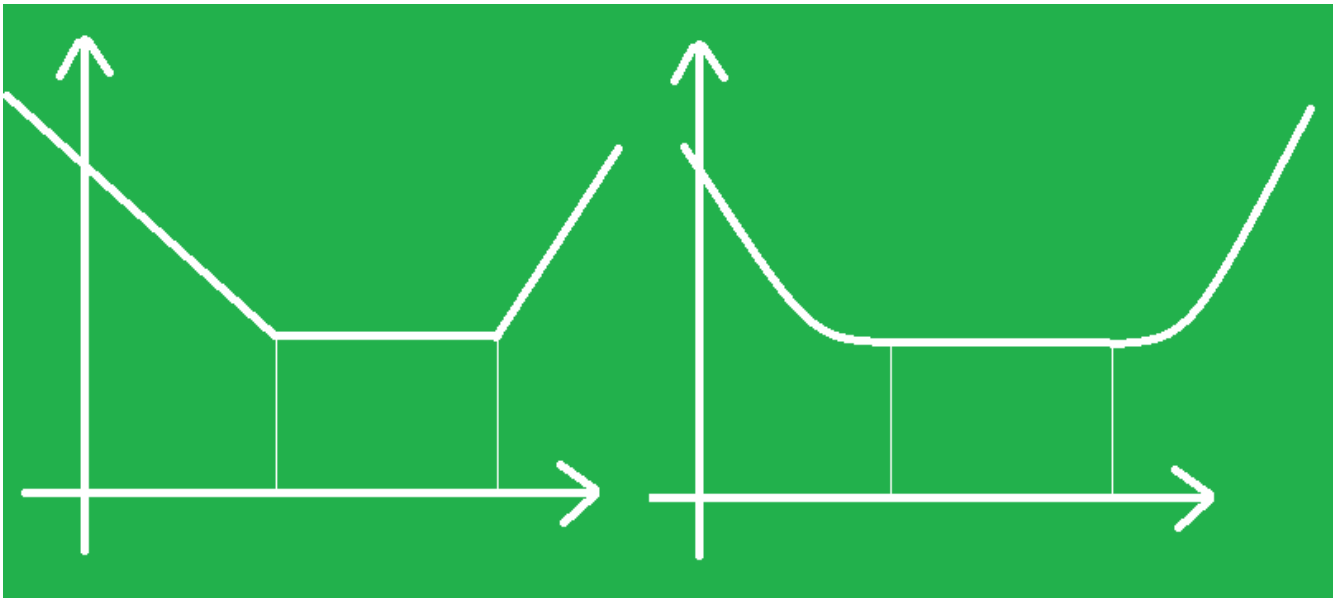
Oui si elle est constante...
Mais sinon, non. Et ça sert dans des exercices.
Pourquoi ? C'est très rapide, juste avec des mots et sans formules de dérivation et tout et tout.
Si elle n'est pas constante, il y a au moins une corde non horizontale.
Et hors du segment, elle doit être au dessus de cette corde. Il y a donc un côté (suivant le signe du coefficient directeur de la corde) où elle tend vers l'infini positif.



d- Pouvez vous construire f convexe de minimum 1 atteint à la fois en 2 et en 3 ?

Oui. Il suffit qu'elle soit constante égale à 1 sur tout le segment entre 2 et 3.
On peut proposer $x \longmapsto \left\{ \begin{array}{lll} x-1 & si & x \leqslant 2 \\ 1 & si & 2 < x < 3 \\ x-2 & si & x \geqslant 3 \end{array} \right|$. Elle est affine par morceaux et convexe.
On peut même demander à ce qu'elle soit dérivable. Même en 2 et 3 : $x \longmapsto \left\{ \begin{array}{lll} (x-2)^2+1 & si & x \leqslant 2 \\ 1 & si & 2 < x < 3 \\ (x-2)^2+1 & si & x \geqslant 3 \end{array} \right|$.
Le plus fort sera de réussir à construire des applications C^∞ avec un plateau sur tout un segment tel que $[2,\ 3]$ ici.

Et bien sûr il y avait la réponse (solution de facilité, donc pur matheux) : f est constante égale à 1. Elle a bien un minimum global 1 atteint en 2 et 3 et pas que...).



e- Si le graphe de f (dérivable) est toujours au dessus de ses tangentes, peut on déduire que f est convexe.

Déjà, si on parle de tangentes, f est dérivable, c'est un fait. Reste à savoir si sa dérivée est croissante.
Question finalement pas si évidente à prouver...
On va montrer l'inégalité des trois cordes pour tour triplet $(a,\ b,\ c)$ avec $a < b < c$.

Le graphe est au dessus de la tangente en b. On a donc $\boxed{f(a) \geqslant f'(b).(a-b)+f(b)}$
$\boxed{f(c) \geqslant f'(b).(c-b)+f(b)}$

On tient compte des signes pour faire passer de l'autre coté ($c-b$ est positif mais $a-b$ est négatif) :
$\boxed{\frac{f(b)-f(a)}{b-a} \leqslant f'(b)}$
$\boxed{\frac{f(c)-f(b)}{c-b} \geqslant f'(b)}$

On glisse $f'(b)$ au milieu : $\frac{f(b)-f(a)}{b-a} \leqslant f'(b) \leqslant \frac{f(c)-f(b)}{c-b}$ donc $\frac{f(b)-f(a)}{b-a} \leqslant \frac{f(c)-f(b)}{c-b}$.

On replie en $\begin{vmatrix} 1 & 1 & 1 \\ a & b & c \\ f(a) & f(b) & f(c) \end{vmatrix} \geqslant 0$ et on développe en n'importe laquelle des trois inégalités sur les cordes.

On a établi la convexité de f.

Il y a donc équivalence entre « convexe » et « au dessus de ses tangentes », en tout cas dans le cas dérivable.

Lycee Charlemagne	MPSI2	Annee 2023/24

Villes

Le but de cette épreuve (filière PSI, mais grand concours) est de décider s'il existe, entre deux villes données, un chemin passant par exactement k villes intermédiaires distinctes, dans un plan contenant au total n villes, reliées par m routes. L'algorithme d'exploration naturel s'exécute en un temps $O(n^k.m)$. L'objectif est d'obtenir un algorithme qui s'exécute en un temps $O(f(k).n.(n+m))$, qui croît polynomialement en la taille $n+m$ du problème, quelle que soit la valeur de k demandée.

La complexité ou temps d'exécution d'un programme Π (fonction ou procédure [5]) est le nombre d'opérations élémentaires (additions, multiplications, affectations, tests...) nécessaires à l'exécution de Π. Lorsque cette complexité dépend de plusieurs paramètres n, m et k, on dira que Π a une complexité $O(\phi(n,m,k))$ lorsqu'il existe quatre constantes absolues A, n_0, m_0 et k_0 telles que la complexité de Π soit inférieure ou égale $A.\phi(n,m,k)$ pour tout $n \geqslant n_0$, $m \geqslant m_0$ et $k \geqslant k_0$. Lorsqu'il est demandé de préciser la complexité d'un programme, le candidat devra la justifier si elle ne se traduit pas directement de la lecture du code.

On suppose qu'on dispose d'une fonction `CreerTableau(n)` *qui crée un tableau de taille* `n`*, indexé de* `0` *à* `n-1` *(les valeurs contenues dans le tableau initialement sont arbitraires [6]. L'instruction* `b=CreerTableau(n)` *créera/affectera un tableau de taille* `n` *dans la variable* `b`*. On pourra créer des tableaux de tableaux : exemple* `a = CreerTableau(p)`

```
for i in range(p) :
....a[i] = CreerTableau(q)
```

On accède par `a[i][j]` *à la case d'indice j du tableau d'indice* `i` *dans le tableau ainsi créé.*

5. *les fonctions renvoient une valeur (entier, réel, tableau, booléen...) ; les procédures modifient des variables globales, mais ne renvoient rien*
6. *avec Python, des* `None`*, qui n'occupent pas de place en mémoire, dans les notations du devoir, de simples* `*`

On souhaite stocker en mémoire une liste non ordonnées d'au plus n entiers sans redondance (aucun entier n'apparaît plusieurs fois). Nous nous proposons d'utiliser un tableau `liste` *de longueur* $n+1$ *tel que* `liste[0]` *contient le nombre d'éléments de la liste (comme en Pascal ?)*
`liste[i]` *contient le* i^{eme} *élément de la liste (non triée) pour* $1 \leqslant i \leqslant liste[0]$ [7]*. Il y a des exemples plus loin, profitez.*
Sinon, vous l'aurez compris, nulle part vous n'utiliserez **`append`***...*

Écrivez une fonction **`CreerListeVide(n)`** qui crée, initialise et renvoie un tableau de longueur **`n+1`** correspondant à la liste « vide » ayant une capacité totale de n éléments.

Écrivez une fonction **`EstDansListe(liste, x)`** qui renvoie **`True`** si l'élément **`x`** apparaît dans la liste représentée par le tableau **`liste`**, et renvoie **`False`** sinon.
■ Quelle est la complexité de votre fonction dans le pire cas en fonction du nombre maximal n d'éléments de la liste.

Écrivez une procédure **`AjouteDansListe(liste, x)`** qui modifie de façon appropriée le tableau **`liste`** pour y ajouter **`x`** si x n'appartient pas déjà à la liste, et ne fait rien sinon.
■ Quel est le comportement de votre procédure si la liste est plein initialement (on ne demandera pas de traiter ce cas).
■ Quelle est la complexité en temps de votre procédure dans le pire cas en fonction du nombre maximal d'éléments de la liste ?

Un plan P est défini par un ensemble de n villes numérotées de 1 *à n et un ensemble de m routes (toutes à double sens) reliant chacune deux villes ensemble. On dira que deux villes x et y sont voisines lorsqu'elles sont reliées par une route, ce que l'on notera* $x \sim y$*. On appellera chemin de longueur k toute suite de villes* $v_1, v_2 \dots v_k$ *telle que* $v_1 \sim v_2 \sim \dots \sim v_k$*. On représentera les villes d'un plan par des ronds contenant leur numéro et les routes par des traits reliant les villes voisines* [8]*.*
Un plan P est donc un tableau `plan` *où*

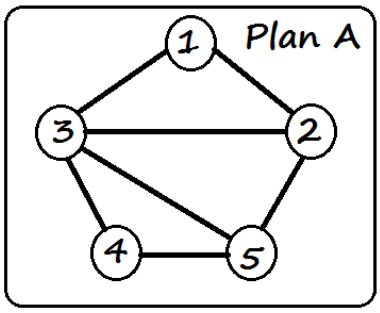
- `plan[0]` *contient un tableau à deux éléments :* `plan[0][0] = n` *(contient le nombre de villes du plan)*
  `plan[0][1] = m` *(contient le nombre de routes du plan)*
- *pour chaque ville x de* $\{1, 2, \dots n\}$, `plan[x]` *contient un tableau à n éléments représentant la liste à au plus* $n-1$ *éléments des villes voisines de la ville* `x` *(ordre de lecture sans importance).*

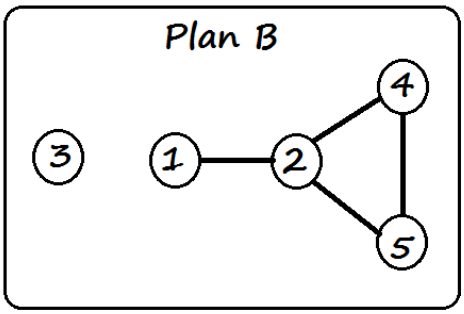
L'un des trois plans du dessin est représenté par le tableau ci contre

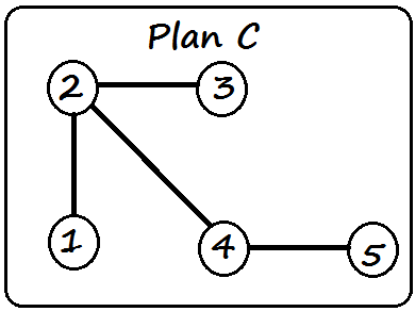
```
plan=[  [5,  4],
        [1,  2,  *,  *,  *  ],
        [3,  4,  1,  5,  *  ],
        [0,  *,  *,  *,  *  ],
        [2,  2,  5,  *,  *  ],
        [2,  4,  2,  *,  *  ]]
```

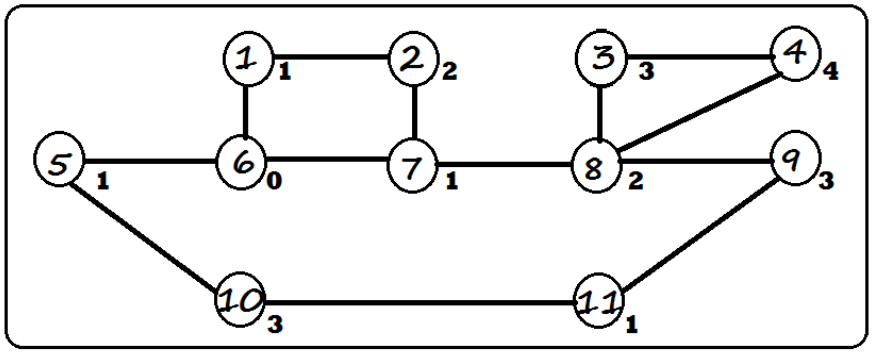
Donnez le tableau des deux autres.

7. *ici pas de décalage pythonien, la case d'indice* 0 *est prise*
8. *comme si vous aviez voulu faire le contraire !*









Graphe coloré.
A côté de chaque ville est indiquée sa couleur.
On cherche donc ici un chemin de la ville [ ] à la ville [ ].

Écrivez une fonction **`CreePlanSansRoute(n)`** qui crée, remplit et renvoie le tableau de tableaux correspondant au plan à **`n`** villes n'ayant aucun route.

Écrivez une fonction **`EstVoisine(plan, x, y)`** qui renvoie **`True`** si les villes **`x`** et **`y`** sont voisines dans le plan codé par **`plan`**, et **`False`** sinon.
Vous avez évidemment le droit d'utiliser les procédures déjà créées.

Écrivez une procédure **`Ajoute(plan, x, y)`** qui modifie le tableau **`plan`** pour ajouter une route entre les deux villes **`x`** et **`y`** (supposées distinctes, oui) si elle n'était pas déjà présente, et ne fait rien sinon.
■ Y a-t-il un risque de dépassement de la capacité des listes ?

Écrivez une procédure **`AfficheToutesLesRoutes(plan)`** qui affiche la liste des routes codées par le tableau plan, chaque route ne devant être mentionnée qu'une fois. Par exemple, (1-2), (2-4), (2-5), (4-5) pour le plan indiqué et retenu plus haut.
■ Quelle est la complexité en temps de votre procédure dans le pire des cas ?

Recherche d'un chemin arc en ciel.

Étant données deux villes distinctes s et t, nous rechercherons un chemin de s à t, passant par exactement k villes toutes distinctes. L'objectif de cette partie est de la suivante est de construire une fonction qui va détecter en un temps linéaire en n.(n+m) l'existence d'un tel chemin avec une probabilité indépendante de la taille $n+m$ du plan.
Le principe de l'algorithme est d'attribuer à chaque ville de $\{1, 2, \dots n\} - \{s, t\}$ une couleur aléatoire codée par un entier aléatoire uniforme $couleur[i] \in \{1, \dots k\}$ stocké dans un tableau **`couleur`** *de taille $n+1$ (la case 0 n'est pas utilisée). Les villes s et t reçoivent respectivement les couleurs spéciales 0 et $k+1$. L'objectif de cette partie est d'écrire une procédure qui détermine s'il existe un chemin de longueur $k+2$ allant de s à t dont la j^{ieme} ville intermédiaire a reçu la couleur j. dans l'exemple de la figure suivante, le chemin $6 \sim 7 \sim 8 \sim 3 \sim 4$ de longueur $5 = k+2$ qui relie $s = 6$ à $t = 4$ vérifie cette propriété pour les couleurs indiquées.*
On suppose l'existence d'une fonction **`EntierAleatoire(k)`** *qui renvoie un entier aléatoire uniforme entre 1 et k. Bref P(EntierAleatoire(k)=c)=1/k pour tout c de 1 à k.*

Écrivez une procédure **`ColoriageAleatoire(plan, couleur, k, s, t)`** qui prend en argument un **`plan`** de n villes, un tableau **`couleur`** de taille n+1, un entier **`k`**, et deux villes **`s`** et **`t`** de $\{1, 2, \dots n\}$ et remplit le tableau **`couleur`** avec une couleur aléatoire uniforme dans $\{1, \dots k\}$ choisie indépendamment pour chaque ville de $\{1, \dots n\} - \{s, t\}$ et les couleurs 0 et $k+1$ pour s et t respectivement.

Nous cherchons maintenant à écrire une fonction qui calcule l'ensemble des villes de couleur c voisines d'un ensemble de villes donné. dans l'exemple de la figure 2, l'ensemble des villes de couleur 2 voisines des villes $\{1, 5, 7\}$ est $\{2, 8\}$.

Écrivez une fonction **`VoisinesDeCouleur(plan, couleur, i, c)`** qui crée et renvoie le tableau codant la liste sans redondance des villes de couleur **`c`** voisines de la ville **`i`** dans le tableau **`plan`** colorié par le tableau **`couleur`**.

Écrivez une fonction **`VoisinesDeLaListeDeCouleur(plan, couleur, liste, c)`** qui crée et renvoie un tableau codant la liste sans redondance des villes de couleur **`c`** voisines d'une des villes présentes dans la liste sans redondances **`liste`** dans le plan **`plan`** colorié par la table **`couleur`**.
■ Quelle est la complexité de votre fonction dans le pire cas en fonction de n et m ?

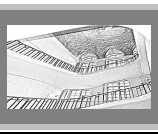
Écrivez une fonction **`ExisteCheminArcEnCiel(plan, couleur, k, s, t)`** qui renvoie **`True`** si il existe dans le plan **`plan`** un chemin $s \sim v_1 \sim \ldots \sim v_k \sim t$ tel que **`couleur[`**v_j**`]=j`** pour tout j de $\{1, \ldots k\}$ et renvoie **`False`** sinon.
■ Quelle est la complexité de votre fonction dans le pire cas en fonction de k, n et m ?

---

*Si les couleurs des villes sont choisies aléatoirement et uniformément dans $\{1, \ldots k\}$, la probabilité que j soit la couleur de la j^{ieme} ville d'une suite fixée de k villes vaut $1/k$, indépendamment pour tout j. Ainsi, étant données deux villes distinctes **`s`** et **`t`**, s'il existe dans le plan **`plan`** un chemin de s à t passant par exactement k villes intermédiaires toutes distinctes et si le coloriage couleur est choisi aléatoirement conformément à la procédure **`ColoriageAleatoire(plan, couleur, k, s, t)`**, la procédure **`ExisteCheminArcEnCiel(plan, couleur, k, s, t)`** répond True avec probabilité au moins k^{-k} ; et répond toujours **`False`** sinon. Ainsi, si un tel chemin existe, la probabilité qu'une parmi k^k exécutions indépendantes de **`ExisteCheminArcEnCiel`** réponde **`True`** est supérieure ou égale à $1-(1-k^{-k})^{k^k} = 1-\exp\left(k^k.\ln(1-k^{-k})\right) \geqslant 1-\frac{1}{e} > 0$ (admis).*

Écrivez une fonction **`ExisteCheminSimple(plan, k, s, t)`** qui renvoie **`True`** avec probabilité au moins $1-\frac{1}{e}$ s'il existe un chemin de **`s`** à **`t`** passant par exactement **`k`** villes intermédiaires toutes distinctes dans le plan **`plan`** et renvoie toujours **`False`** sinon.
■ Quelle est sa complexité en fonction de k, n et m dans le pire des cas ? Exprimez la sous la forme $O(f(k).g(n,m))$ pour f et g bien choisies.

Expliquez comment modifier votre programme pour renvoyer le chemin s'il est détecté avec succès.



Liste sans redondance. TD28

```
def CreerListeVide(n) :
....L = CreerTableau(n+1)
....L[0] = 0
....return L
```

Le tableau est de taille $n+1$ pour avoir en première case le nombre d'éléments.
On initialise le nombre de termes à 0. C'est après, quand on agrandira la liste qu'il faudra modifier l'élement de position 0.

```
def EstDansListe(liste, x) :
....for i in range(1, liste[0]+1) :
........if liste[i] == x :
............return True
....return False
```

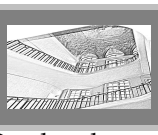
`range(1, liste[0]+1)` car on n'explore pas **`liste[0]`** qui contient juste une information, et aussi parce que ce n'est pas la peine de chercher plus loin que la fin de liste.
Surtout pas **`....if liste[i] == x :`** sinon c'est dès le premier test que vous sortez.
`........return True`
`....else :`
`........return False`
Si vous faites ça, par exemple avec **`liste = [4, 2, 3, 5, 1]`** (de longueur 4 et d'éléments 2, 3, 5 et 1), si vous lancez la procédure pour **`x=5`**, le premier et seul test fait **`2 == 5`**, qui est faux, vous sortez avec **`False`** et vous n'allez pas au delà...
■Le complexité au pire est $O(n)$ où n est le nombre d'éléments. Atteinte si l'élément n'est pas dans la liste, ou

éventuellement si il y est mais que c'est lui le dernier.

```
def AjouteDansListe(liste, x) :
....if not(EstDansListe(liste, x)) :
........liste[0] += 1
........liste[liste[0]] = x
```

On n'oublie pas de dire qu'il y a un élément de plus. Et on sait où l'insérer, c'est à ça que sert ce premier élément `liste[0]` !.
On peut aussi parcourir la liste, sortir sans rien faire si l'élément n'est pas dans la liste, et sinon l'ajouter à la fin.

■Le risque est « si la liste est pleine et *x* n'en fait pas déjà partie ». On colle alors x dans un emplacement qui n'existe pas...



Les plans. TD28

On cherche un plan à cinq villes et quatre routes. Il n'y a que le plan B.
Et effectivement, la ville 1 a un voisin, c'est 2 : `[1, 2, *, *, *]`
la ville 2 a trois voisins : 1, 4 et 5 : `[3, 4, 1, 5, *]` (l'ordre importe peu)
la ville 3 n'a pas de voisin : `[0, *, *, *, *]`
la ville 4 a deux voisins : 2 et 5 : `[2, 2, 5, *, *]`
la ville 5 a deux voisins, je vous laisse compléter

```
planA = [  [5,  7  ],
           [2,  2,  3,  *,  *  ],
           [3,  1,  3,  5,  *  ],
           [4,  1,  2,  4,  5  ],
           [2,  3,  5,  *,  *  ],
           [3,  2,  3,  4,  *  ]]
             ↑   =   =   =   =
```

et

```
planC = [  [5,  4  ],
           [1,  2,  *,  *,  *  ],
           [3,  1,  4,  3,  *  ],
           [1,  2,  *,  *,  *  ],
           [2,  5,  2,  *,  *  ],
           [1,  4,  *,  *,  *  ]]
```

La liste marquée d'une flèche indique combien chaque ville a de voisins.
Elle ne peut pas en avoir plus que quatre. Sur chaque ligne, les éléments peuvent être cités dans un autre ordre que celui que j'ai indiqué ici.

```
def CreerPlanSansRoute(n) :
....L = CreerTableau(n+1)
....L[0] = CreerTableau(2)
....L[0][0], L[0][1] = n, 0
....for i in range(1, n+1) :
.......L[i] = CreerListeVide(n-1)
....return L
```

`L[0]` c'est `[n, 0]` car il y aura n listes et aucune route (ça va venir).
Et pour chaque ville, la liste est vide.
Dur d'écrire ça sans nos réflexes **`append`**. On doit tout fabriquer des tableaux avec des `*`, puis remplacer chaque `*` par une liste créée sur le même principe.
Les correcteurs X-ENS ont ils accepté `L[0] = [n, 0]` à la place de `L[0] = CreerTableau(2)` `L[0][0], L[0][1] = n, 0` ?

```
def EstVoisine(plan, x, y) :
....return EstDansListe(plan[x], y)
```

■ On espère que le tableau est bien rempli et que `EstDansListe(plan[y], x)` retournera le même booléen.
On suppose que x et y sont des index convenables, pas trop grands.

```
def Ajoute(plan, x, y) :
....if not(EstVoisine(plan, x, y)) :
........plan[0][1] += 1
........AjouteDansListe(plan[x], y)
........AjouteDansListe(plan[y], x)
```

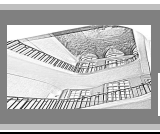
On n'oublie pas qu'il y a une route en plus (et une seule). Et la route est à double sens.
■ On admettra que l'utilisateur ne lance pas la procédure avec x égal à y.
C'est le seul cas où on risque un débordement de liste, car une ville ne peut avoir plus de $n-1$ voisins...

```
def AfficheToutesLesRoutes(plan) :
....n, m = plan[0][0], plan[0][1]
....ToutesLesRoutes = CreerListeVide(m)
....for x in range(1, n+1) :
........NbVoisins = plan[x][0]
........for i in range(1,NbVoisins+1) :
............y = plan[x][i]
............if x < y :
................AjouteDansListe(ToutesLesRoutes, [x,y])
.... for Route in ToutesLesRoutes :
........print(Route)
```

La liste des routes sera, on le sait de longueur m, à condition de ne pas compter les routes deux fois.
On prend les villes une par une (index de `1` à `n`, donc `range(1, n+1)`).
Pour chacune, on va lire la liste de ses voisins, autant en connaître la longueur.
Chaque voisin est `plan[x][i+1]` (ou alors `for i in range(1, NbVoisins+1)`).
Mais pour ne pas compter la route [5, 2] et la route [2, 5], on pense au test `if x < y`.
L'énonce n'est pas clair ici : • on affiche les routes
• on retourne une liste de routes ?
On croise aussi la proposition suivante :

```
def AfficheToutesLesRoutes(plan) :
....n = plan[0][0]
....m = plan[0][1]
....Mot = CreeListeVide(m)
....for x in range(1, n+1) :
........for y in range(x+1, n+1) :
............if EstVoisine(plan, x, y) :
................AjouteDansListe(Mot, "(" + str(x) + "," + str(y) + ")")
....return Mot
```

■ On parcourt n liste. Chacune est de longueur n. On peut craindre une complexité en $n \times n$.
Mais en fait, la liste des voisins de x n'est parcourue que pour son nombre de voisins.
En fait, la somme des longueurs de ces listes est égale à $2 \times m$.
On a donc une complexité en $n + 2 \times m$.
Le facteur 2 nous importe peu : $O(n+m)$.



Coloriage. TD28

```
def ColoriageAleatoire(plan, couleur, k, s, t) :
....n = plan[0][0]
....for i in range(1, n+1) :
........Couleur[i] = EntierAleatoire(k)
....Couleur[s], Couleur[t] = 0, k+1
```

Le tableau couleur est déjà initialisé. On a juste à remplir ses cases.
On colorie chaque ville au hasard.
Et après coup, on rectifie pour les villes x et y.
On pouvait aussi choisir

```
def ColoriageAleatoire(plan, couleur, k, s, t) :
....n = plan[0][0]
....for i in range(1, n+1) :
........if (i == s) :
............Couleur[i]= 0
........elif(i == t) :
............Couleur[i] = k+1
........else :
............Couleur[i] = EntierAleatoire(k)
```

```
def VoisinesDeCouleur(plan, couleur, i, c) :
....n = plan[0][0]
....L = CreerListeVide(n)
....NbVoisins = plan[i][0]
....for j in range(1,NbVoisins+1) :
........if couleur(plan[i][j]) == c :
............AjouteDansListe(L, plan[i][j])
....return L
```

La liste sera de longueur maximal n, le nombre de villes (qu'il faut aller chercher).
On prend une à une les villes voisines (les `plan[i][j]` avec `j` dans le bon range).
On regarde si leur couleur est bonne. Et si oui, on ajoute dans la liste. En utilisant la procédure déjà créée...
Variante :

```
def VoisinesDeCouleur(plan, couleur, i, c) :
....n = plan[0][0]
....voisines_de_i_de_couleur_c = CreerListeVide(n)
....nb_voisines_de_i = plan[i][0]
....for k in range(1, nb_voisines_de_i+1) :
........j = plan[i][k]
........if i != j and couleur[j] == c :
............AjouteDansListe(voisines_de_i_de_couleur_c, j)
....return voisines_de_i_de_couleur_c
```

```
def VoisinesDeLaListeDeCouleur(plan, couleur, liste, c) :
....n = plan[0][0]
....L = CreerListeVide(n)
....NbVilles = liste[0]
....for i in range(1, NbVilles+1) :
........LaVille = liste[i]
........NbVoisins = plan[LaVille][0]
........for j in range(1, NbVoisins+1) :
............LeVoisin = plan[LaVille][j]
............if Couleur[LeVoisin] == c :
................AjouteDansListe(L, LeVoisin)
....return L
```

Avec `AjouteDansListe`, on ne met pas plusieurs fois une ville déjà prise.
Ça peut donner aussi

```
def VoisinesDeLaListeDeCouleur(plan, couleur, liste, c) :
....L = CreerListeVide(plan[0][0])
....for i in range(1, liste[0]+1) :
........for j in range(1, plan[liste[i]][0]+1) :
............if Couleur[plan[liste[i]][j]] == c :
................AjouteDansListe(L, plan[liste[i]][j])
....return L
```

Comptez bien les crochets à chaque fois, le compte est bon.
J'appelle ça une version « one-liner ». On ne crée pas de variable intermédiaire, on imbrique des fonctions des unes dans les autres.

```
def ExisteCheminArcEnCiel(plan, couleur, k, s, t):
....n = plan[0][0]
....Liste = CreerListeVide(n)
....AjouteDansListe(Liste, s)
....for c in range(1, k+2):
........Liste = VoisinesDeLaListeDeCouleur(plan, couleur, Liste, c)
........if Liste[0] == 0:
............return False
....return EstDansListe(Liste, t)
```

On crée une liste des villes. Au départ, elle ne contient que s.
A chaque étape, on regarde quelles nouvelles villes on peut atteindre.
Invariant de boucle : à l'étape c, on a les villes qu'on peut atteindre en k étapes.
Si à un moment on ne peut plus rien atteindre, autant s'arrêter.
Sinon, à la fin, on regarde si la ville objectif **t** est dans la liste des villes atteignables.
On retourne le booléen **EstDansListe(Liste, t)** car on fait de l'informatique.
Si on fait n'importe quoi, on termine par

```
if EstDansListe(Liste, t):
....return True
else:
....return False
```

■ Il y a k boucles (au pire) qui sollicitent $n.(n+m)$ opérations.
La complexité est en $O(k.n.(n+m))$.

J'ai croisé la proposition suivante de la part de collègues. Elle me laisse perplexe :

```
def ExisteCheminArcEnCiel(plan, couleur, k, s, t):
....n = plan[0][0]
....# On se lance, en partant de s
....liste = CreerListeVide(n) # Sommets de couleur 0
....AjouteDansListe(liste, s) # Sommets de couleur 0
....# Couleur courant 0 (note : c aussi = diamètre de l'exploration courante)
....c = 0
....while not(EstDansListe(liste, t)) and c <= t : # On n'est pas arrivé
........c += 1 # On passe à la couleur d'après
........# On filtre les voisines des villes dans liste qui sont de couleurs c
........bonnes_voisines = VoisinesDeLaListeDeCouleur(plan, couleur, liste,
c)
........# Et ou bien on peut continuer à explorer, ou bien on doit arrêter
........if bonnes_voisines[0] == 0:
............# On doit s'arrêter
............return False
........ else:
............# On peut continuer, et désormais on part des villes suivantes
............liste = bonnes_voisines
....# On a t dans la liste, donc on a un chemin partant de s jusqu'à t
....return True
```

Qu'est ce qui me gène ? Cette fonction ne fait pas intervenir **k**...
Mais c'est dû à une petite faute de frappe que vous devriez détecter...

J'ai aussi croisé une correction de cet exercice avec la fonction (à peu près) suivante :

```
def ExisteCheminArcEnCiel(plan, couleur, k, s, t):
....n = plan[0][0]
....Liste = CreerListeVide(n)
....AjouteDansListe(Liste, s)
....for c in range(1, k+2):
........Liste = VoisinesDeLaListeDeCouleur(plan, couleur, Liste, c)
........if Liste[0] == 0:
............return False
....return True
```

Cette fois, c'est t qui n'avait aucun rôle. On regardait bien si il existait un chemin arc en ciel, mais on ne regardait pas si il allait bien de *s* à *t*.

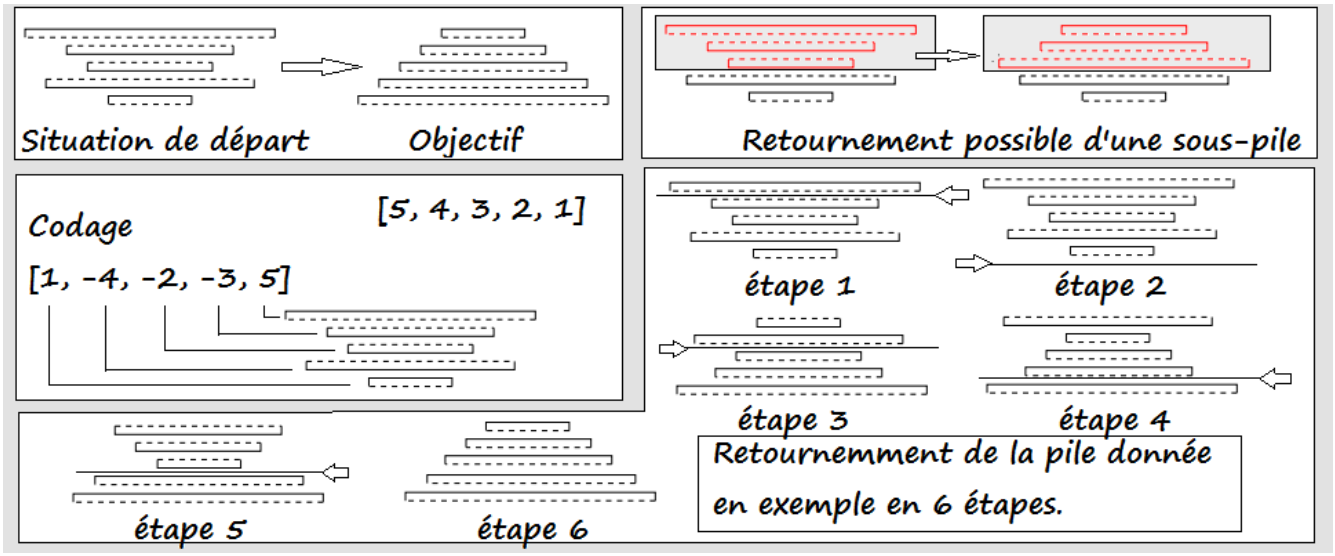
```
def existeCheminSimple(plan, k, s, t) :
....n = plan[0][0]
....nombre_executions = k ** k
....for i in range(nombre_executions) :
........# On génère un coloriage aléatoire
........couleur = CreerTableau(n+1) # Tableau vide
........ColoriageAleatoire(plan, couleur, k, s, t)
........# On espère qu'il convient pour trouver un chemin arc-en-ciel
........if existeCheminArcEnCiel(plan, couleur, k, s, t) :
............return True
.... return False
```

∘32∘ On se propose d'étudier un algorithme de tri sur des listes, dit « algorithme du crêpier ». Oui, du marchand de crêpes.
Le crêpier fabrique des crêpes qui ont des diamètres tous différents, et de plus chacune a deux faces : une face dorée, et une face blanche, pas assez cuite. Il les a posées dans le désordre, et chacune dans n'importe quel sens.

Son objectif : une belle pile de crêpes, toutes orientées dans le même sens, face blanche vers le haut, de la plus grande à la plus petite.

Ce qu'il a le droit de faire : attraper le dessus du paquet de crêpes, sur une certaine épaisseur, puis retourner ce sous-paquet.
(les *n* crêpes de ce paquet sont alors changées de sens, et la sous-pile elle même est renversée).

Sur le dessin, on donne un exemple de retournement (optimal ?) d'une pile de cinq crêpes.



Une pile de crêpes sera codée par une liste d'entiers relatifs tous distincts en valeur absolue, de la crêpe du bas à la crêpe du haut. La valeur absolue de l'entier donne son diamètre, et le signe donne son orientation (deux bonnes raisons qu'il n'y ait pas de crêpe codée 0).
La pile finale devra donc être faite d'entiers positifs, du plus grand au plus petit.

I∼0) Donnez le codage de chacune des six piles de l'exemple représenté dans le cadre ci dessus.

I∼1) Écrivez une procédure `Melange` qui prend en entrée un entier n et retourne une pile de n crêpes, désordonnée.
Exemple : `Melange(5)` pourra donner `[1, -4, -2, -3, 5]`.

Rappel : le module **randrange** contient les fonctions **randrange**, **random** et **shuffle**.
shuffle(L) mélange une liste L : exemple `L = [3, 6, 4, 2, 0]`
`shuffle(L)`
`print(L)` : `[0, 3, 2, 4, 6]`
randrange(a,b) donne un entier entre a (inclus) et b (exclu)
random() donne un flottant au hasard entre 0 et 1.
Un bonus si vous écrivez une procédure qui n'utilise pas **shuffle** mais juste des **randrange**.

I∼2) Écrivez une procédure **Test** qui prend en entrée une liste de crêpes L et retourne un booléen pour dire si la pile est triée, avec ses crêpes dans le bon sens.
Exemple : `Test( [6, 4, 3, 2, 1])` donne **True**,
`Test( [7, -5, 4, 3, 2, -1])` donne **False**,
`Test([7, 4, 5, 3, 2, 1])` donne **False**.

I∼3) Écrivez une procédure **Retourne** qui prend en entrée une liste L et un index n et retourne la pile de crêpes L retournée à partir de l'indice n.
Exemple : `Retourne([6, 7, -5, 4, 2, -3], 3)` donne `[6, 7, -5, 3, -2, -4]`.

Rappel : on peut couper une liste d'un indice a (inclus) à un indice b (exclu), de plus une option indique avec quel pas on avance dans la liste (par défaut, c'est bien sûr 1).
Exemple : `L = [1, 3, 5, 7, 8, 2, 6, 5, 9]`
`L[2 : 5]` donne `[5, 7, 8]`
`L[ : 4]` donne `[1, 3, 5, 7]`
`L[2 : 7 : 2]` donne `[5, 8, 6]`
`L[9 : 1 : -2]` donne `[9, 6, 8, 5]`
`L[1 : 9 : -2]` donne `[ ]`

```
def Scooby(L) :
....M, i = abs(L[0]), 0
....for k in range(len(L)) :
........if abs(L[k]) > M :
............M, i = L[k], k
....return i
```

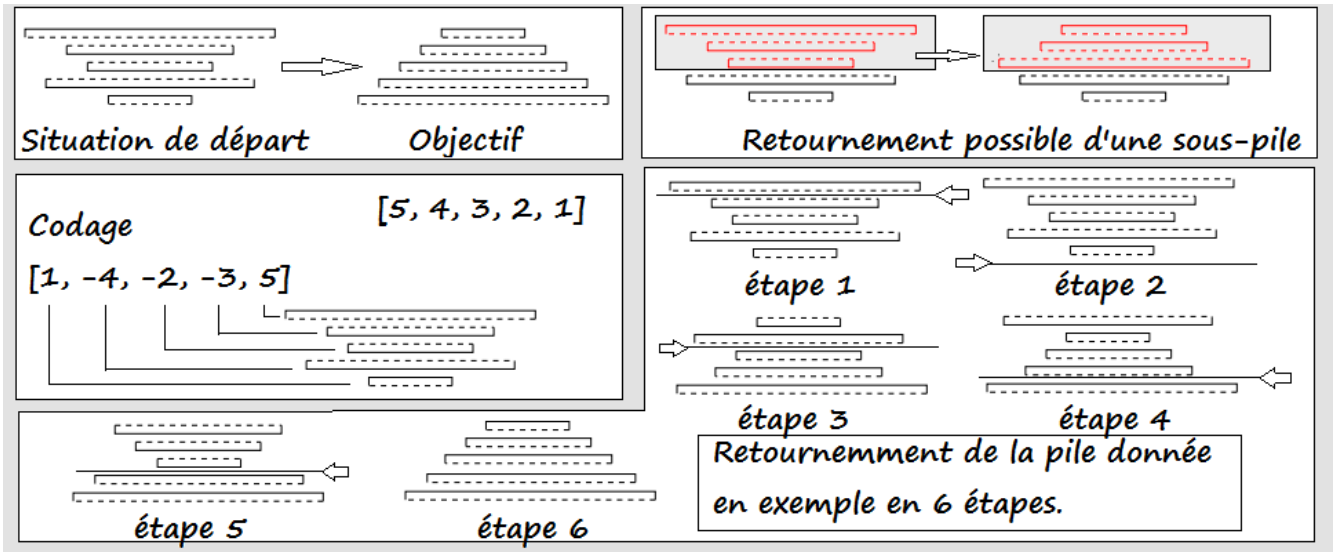
I∼4) Que fait cette procédure :
Que retournera `Scooby([2, 5, -8, 7, 1, 3, 4, -6])` ?

I∼5) Complétez

	[-7, 2, 5, 1, 3, 4, -6]
Retourne(L, 0)	*[6, -4, -3, -1, -5, -2, 7]*
Retourne(L, 6)	*[6, -4, -3, -1, -5, -2, -7]*
Retourne(L,)	*[7, 2, 5, 1, 3, 4, -6]*
Retourne(L,)	*[7, 6, -4, -3, -1, -5, -2]*
Retourne(L, 5)	
Retourne(L,)	*[7, 6, -4, -3, -1, 2, -5]*
Retourne(L,)	*[7, 6, 5, , , ,]*
Retourne(L, 6)	*[7, 6, 5, , , ,]*
	[7, 6, 5, 4, 3, 2, 1]

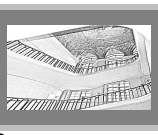
I∼6) Quelle est la pile de cinq crêpes qui sera la plus longue à retourner par algorithme du crêpier, et en combien de retournements ? (la plus courte est la pile déjà triée `[5, 4, 3, 2, 1]`).

I∼7) Écrivez un script qui prend en entrée une liste L et la trie par algorithme du crêpier.



étape 0	étape 1	étape 2	étape 3	étape 4
`[1, -4, -2, -3, 5]`	`[1, -4, -2, -3, -5]`	`[5, 3, 2, 4, -1]`	`[5, 3, 2, 1, -4]`	`[5, 4, -1, -2, -3]`
`[1, -4, -2, -3] + [5]`	`[ ] + [1, -4, -2, -3, -5]`	`[5, 3, 2] + [4, -1]`	`[5] + [3, 2, 1, -4]`	`[5, 4] + [-1, -2, -3]`
			étape 5	`[5, 4, 3, 2, 1]`

Une ligne a été ajoutée pour comprendre où on coupe.



Mélange. TD28

Comme on veut des entiers distincts, on ne va pas se prendre la tête : `range(1, n+1)` (attention, on commence à 1 et on doit donc terminer à n inclus).
On pourrait commencer avec `[k for k in range(1, n+1)]`.

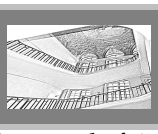
On veut un signe plus ou moins : `randrange(-1, 2, 2)` prend un entier entre -1 et 2 (exclu) mais avec pas de 2 : bref : -1 ou 1.
On reprend donc avec `[randrange(-1, 2, 2)*k for k in range(1, n+1)]`.

On veut la mélanger ? Un petit coup de `shuffle`.

```
def Melange(n) :
....L = [randrange(-1, 2, 2)*k for k in range(1, n+1)]
....shuffle(L)
....return(L)
```

```
def Melange(n) :
....L = [ ]
....for k in range(1, n+1) :
........if randrange(2) == 0 :
............L.append(k)
........else :
............L.append(-k)
....shuffle(L)
....return(L)
```

```
def Melange(n) :
....L, LL = [ ], [ ]
....for k in range(1, n+1) :
........if randrange(2) == 0 :
............L.append(k)
........else :
............L.append(-k)
....while len(L) > 0 :
........a = L.pop(randrange(len(L))
........LL.append(a)
....return(LL)
```



Test. TD28

On peut le faire en deux fois. On teste d'abord si toutes les crêpes sont dans le bon sens

```
for crepe in L :
....if crepe =< 0 :
........return(False)
```

```
for crepe in L :
....if crepe =< 0 :
........return(False)
....else :
........return(True)
```

Et surtout pas

En effet, avec cette option, vous sortez dès la première crêpe étudiée, avec `True` ou `False`, sans étudier les autres crêpes.
Ce qu'il faut, c'est répondre `False` si on croise une crêpe dans le mauvais sens, c'est tout.

```
for k in range(len(L)) :
....if L[k] =< 0 :
........return(False)
```

Sinon, vous pouvez aussi utiliser c'est pareil.

Ensuite, on regarde si chaque crêpe est plus grande que la crêpe suivante.

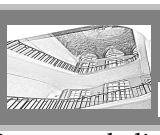
```
for k in range(len(L)-1) :
....if L[k+1] > L[k] :
........return(False)
```

Attention, même chose, on ne met le `return(True)` que si on a parcouru toute la liste sans erreur.
De plus, comme on compare `L[k]` et `L[k+1]`, il faut arrêter `k` un index plus tôt que d'habitude.

```
def Test(L) : #list of int -> boolean
....for crepe in L :
........if crepe =< 0 :
............return(False)
....for k in range(len(L)-1) :
........if L[k+1] > L[k] :
............return(False)
....return(True) #tout s'est bien passé
```

Variante en une fois (on teste « positif et plus grand que le suivant » dont la négation est connue, et il reste le dernier à regarder).

```
def Test(L) :
....for k in range(len(L)-1) :
........if L[k] =< 0 or L[k+1} > L[k] :
............return(False)
....return(L[-1]>0) #reste à tester juste si le dernier est positif
```



Retournement. TD28

On coupe la liste en deux justement à partir de l'indice n.
Le paquet du bas ne contiendra que les termes de 0 (valeur par défaut) à n (exclu).
Le paquet du haut contiendra les termes de l'indice n (inclus) jusqu'à la fin (valeur par défaut).

```
Bas = L[0 : n]
Haut = L[n : len(L)]
```

```
Bas = L[ : n]
Haut = L[n : ]
```

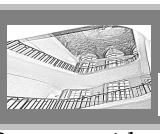
On retourne `Haut` soit avec méthode `reverse`,
soit par parcours avec un pas de `-1` : `Haut = Haut[ : : -1]` (si si, ça marche).
On change les signes de tout le monde : `[-x for x in Haut]`.
On recolle les deux : `Bas + Haut`.

```
def Retourne(L) :
....Bas, Haut = L[ : n], L[n : ]
....tuaH = [-x for x in Haut[ : : -1]]
....return(Bas + tuaH)
```

On peut aussi recopier la fin de la liste et changer ensuite les valeurs des éléments de la liste `L`.

```
def Retourne(L, n) : #list, int -> list
....#retourne la liste L à partir de la position i, en changeant le signe de chaque crêpe
....Pile = L[n : ] #copie de la fin de la liste
....for k in range(n, len(L)) :
........L[k] = -Pile[n-k-1] #remplacement d'un élément de L par l'opposé de l'élément de Pile lu depuis la fin
....return(L)
```

On vérifie : pour `k = i`, on va chercher l'élément d'indice `-1` dans `Pile`, c'est le dernier.
pour `k = i+1`, on va chercher l'élément d'indice `-2` dans `Pile` : l'avant dernier
pour `k = len(L)-1`, on va chercher l'élément d'indice `n-Len(L)` dans `Pile`, et sachant que `Pile` a `Len(L)-n` éléments, c'est le premier.
Oui étrangement, `L[len(L)]` n'existe pas (`index out of range` trop long) mais `L[-len(L)]` existe, et c'est `L[0]`.



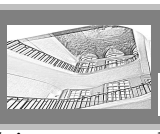
Procédure Scooby. TD28

Cette procédure ressemble fort à la recherche du maximum d'une liste.

```
def Vera(L) : #list of *** -> ***
....M = L[0] #on initialise
....for k in range(len(L)) : #on parcourt la liste
........if L[k] > M : #si on trouve un nouveau record
............M = L[k] #c'est lui qu'on enregistre
....return(M)
```

Ce qui change ici : on met des valeurs absolues. On cherche donc le plus grand terme en valeur absolue.
De plus, on mémorise le record, mais aussi son index k. Et c'est lui qu'on retourne.
On cherche donc l'index du terme le plus grand en valeur absolue.

`Scooby([2, 5, -8, 7, 1, 3, 4, -6])` va donc retourner 2 puisque c'est `L[2]` qui est le plus grand en valeur absolue.

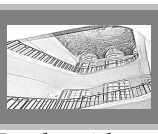


Un exemple. TD28

J'ajoute une colonne pour la compréhension

		[-7, 2, 5, 1, 3, 4, -6]
Retourne(L, 0)	[ ] + [-7, 2, 5, 1, 3, 4, -6]	[6, -4, -3, -1, -5, -2, 7]
Retourne(L, 6)	[6, -4, -3, -1, -5, -2] + [7]	[6, -4, -3, -1, -5, -2, -7]
Retourne(L, 0)	[ ] + [6, -4, -3, -1, -5, -2, -7]	[7, 2, 5, 1, 3, 4, -6]
Retourne(L, 1)	[7] + [2, 5, 1, 3, 4, -6]	[7, 6, -4, -3, -1, -5, -2]
Retourne(L, 5)	[7, 6, -4, -3, -1} + [-5, -2]	[7, 6, -4, -3, -1, 2, 5]
Retourne(L, 6)	[7, 6, -4, -3, -1, 2} + [5]	[7, 6, -4, -3, -1, 2, -5]
Retourne(L, 2)	[7, 6] + [-4, -3, -1, 2, -5]	[7, 6, 5, -2, 1, 3, 4]
Retourne(L, 6)	[7, 6, 5, -2, 1, 3] + [4]	[7, 6, 5, -2, 1, 3, -4]
Retourne(L, 3)	[7, 6, 5] + [-2, 1, 3, -4]	[7, 6, 5, 4, -3, -1, 2]
Retourne(L, 4)	[7, 6, 5, 4] + [-3, -1, 2]	[7, 6, 5, 4, 2, 1, 3]
Retourne(L, 6)	[7, 6, 5, 4, 2, 1] + [3]	[7, 6, 5, 4, 2, 1, -3]
Retourne(L, 4)	[7, 6, 5, 4] + [2, 1, -3]	[7, 6, 5, 4, 3, -1, -2]
Retourne(L, 5)	[7, 6, 5, 4, 3] + [-1, -2]	[7, 6, 5, 4, 3, 2, 1]

Il me semble bien que c'est la pile [-5, -4, -3, -2, -1] qui sera la plus longue à retourner.
Les crêpes sont dans le bon ordre, mais toutes à l'envers.
[-5, -4, -3, -2, -1]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, -5]
[5, -4, -3, -2, -1]
[5, 1, 2, 3, 4]
[5, 1, 2, 3, -4]
[5, 4, -3, -2, -1]
[5, 4, 1, 2, 3]
[5, 4, 1, 2, -3]
[5, 4, 3, -2, -1]
[5, 4, 3, 1, 2]
[5, 4, 3, 1, -2]
[5, 4, 3, 2, -1]
[5, 4, 3, 2, 1]
Mais je m'y prends put être mal. Le nombre de fois où je retourne juste la crêpe du haut du paquet !



Algorithme. TD28

Un algorithme prend forme.
Une partie de la liste est triée, mais pas encore la liste entière.
Regarder ce qu'il reste.
Trouver dans ce qu'il reste la plus grand crêpe.
La placer au dessus du paquet en renversant à partir de sa position.
Si elle est dans le « mauvais sens », retourner juste cette crêpe.
Retourner la pile pour mettre alors la plus grande crêpe à la bonne place.

Par exemple, avec [6, 5, 3, -4, 2, 1].
[6, 5] est déjà trié.
On regarde [3, -4, 2, 1].
Le plus grand élément en valeur absolue est -4, avec index (global) 2.
On renverse la liste initiale à partir de l'indice 2 :

$$[6,5,3,-4,2,1] -> [6,5,3] + [-4,2,1] -> [6,5,3] + [1,-2,4] -> [6,5,3,1,-2,4]$$

On retourne la dernière crêpe avec Renverse(L, 6) :

$$[6,5,3,1,-2,4] -> [6,5,3,1,-2] + [4] -> [6,5,3,1,-2] + [-4] -> [6,5,3,1,-2,-4]$$

On renverse à partir de l'indice 2 : Renverse(L, 2) :

$$[6,5,3,1,-2,-4] -> [6,5] + [3,1,-2,-4] -> [6,5] + [4,2,-1,-3] -> [6,5,4,2,-1,-3]$$