Petits jeux mathématiques

Avant de commencer, dirigez vous vers le site internet

`http://pcsi.kleber.free.fr/IPT/`

et téléchargez le dossier zippé `TP_jeux_eleves.zip`. Stockez-le dans un répertoire `Info` de votre dossier personnel et dézippez-le. Ouvrez votre environnement de travail et sauvegardez un fichier vide sous le nom `TP_jeux_mathematiques.py` pour que les procédures de tests puissent s'effectuer correctement (voir, comme les fois précédents, le fichier `testeur.py` du dossier `TP_jeux`).

Partie I

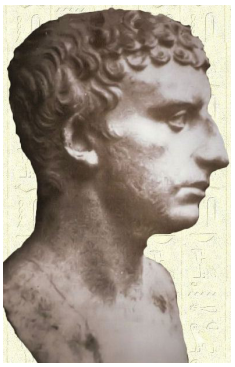
Petite somme

En utilisant deux boucles convenablement imbriquées, calculer

$$\sum_{i=1}^{20}\sum_{j=1}^{20}(i+j)^2$$

Partie II

Le problème de Josèphe



Nous allons étudier une variante d'un problème ancien, à qui on a donné le nom de Flavius Josèphe, célèbre historien du premier siècle.

En l'an 66, dans des circonstances qui sont un peu floues, Flavius Josèphe se serait retrouvé piégé lors de la prise de la citadelle de Jotapata en Galilée par l'armée romaine. Il se retrouve pris au piège et assiégé dans une grotte avec quarante de ses compagnons. Il sont donc en tout 41. Les Romains leur demandent de se rendre, mais ils refusent. Choisissant de mourir plutôt que d'être capturés, les 41 compagnons décidèrent de former un cercle puis, en suivant sa circonférence, de tuer un homme sur trois comptés parmi les survivants, jusqu'à ce qu'il ne reste plus que deux combattants, dont l'un tuerait l'autre puis se suiciderait. Josèphe ne voulait pas de ce suicide absurde. Il en était de même pour un autre compagnon. Josèphe calcula rapidemant où son ami et lui devaient se placer dans le cercle vicieux. Ce problème mathématique, est appelé le problème de Josèphe ou Roulette Romaine.

Nous allons chercher à résoudre ce problème et nous en donnerons des généralisations.

1. Dans notre variante, on commence avec n personnes numérotées de 1 à n sur un cercle. On élimine une personne sur deux jusqu'à ce qu'il n'y ait plus qu'un survivant. Par exemple, si $n = 10$, on élimine dans l'ordre 2, puis $4, 6, 8, 10, 3, 7, 1, 9$. Le survivant est donc 5. L'objectif est de construire un programme qui calcule le numéro $\mathrm{J}(n)$ du survivant.
   a) Calculer — à la main — les positions $\mathrm{J}(4)$ et $\mathrm{J}(5)$ afin de bien comprendre le problème.
   b) Écrire un programme `josephe(n)` qui prend comme paramètre un entier n et qui retourne le numéro $\mathrm{J}(n)$ du survivant.
2. Modifier le programme ci-dessus pour obtenir la solution du problème de Josèphe.
3. Terminons par une dernière variante. Supposons que $2n$ personnes forment un cercle ; les n premières sont « gentilles », tandis que les n dernières sont « méchantes ». Construire un programme qui retourne le plus petit entier m tel que, si on parcourt le cercle en supprimant chaque m^{e} personne, les méchants sont les seuls à être exécutés. (Par exemple, si $n = 3$, l'entier recherché vaut 5 ; si $n = 4$, on peut prendre $m = 30$).

Partie III

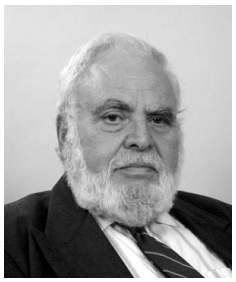
Nombre de zéros

On souhaite savoir combien de 0 terminent l'écriture de 5000!. Pour cela, on va procéder de la manière suivante :

1. Construire une fonction `factoriel(n)` qui à l'appel de l'entier naturel n retourne $n!$.
2. Construire une fonction `nb_zero_factoriel(n)` qui à l'appel de l'entier naturel n retourne le nombre de zéros qui terminent l'écriture décimale de $n!$. Répondre à la question posée.

Partie IV

Suite de Golomb



Solomon Wolf Golomb, né en 1932 à Baltimore, Maryland, est un mathématicien et un informaticien américain à l'origine du graphe de Golomb, des règles de Golomb, de la suite de Golomb et du codage de Golomb (en 1966) utilisé en compression de données en informatique. Il a également introduit les polyominos dans un livre intitulé Polyominoes ; ce livre a inspiré le célèbre jeu Tétris.

La suite de Golomb, que nous noterons $(g_n)_{n\geqslant 1}$, est une suite croissante d'entiers naturels dont le premier terme g_1 vaut 1 et telle que pour tout entier n supérieur ou égal à 1, le n^{e} terme de la suite de Golomb est le nombre d'occurrences de l'entier n dans cette suite.

Les vingt premiers termes de la suite de Golomb sont :

$$1, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 8, \ldots$$

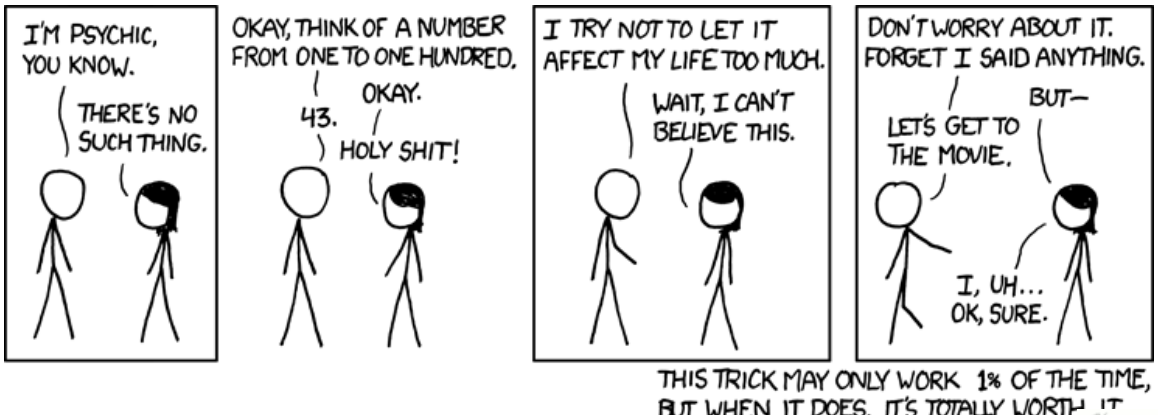
Par exemple, le 7^{e} terme de la suite est 4, donc l'entier 7 apparaît 4 fois dans la suite. Construire une procédure `golomb(n)` qui retourne le n^{e} terme de la suite de Golomb.

Partie V

Multiples en 11...11

On peut montrer que tout entier impair non multiple de 5 possède un multiple de la forme $111\ldots111$ c'est-à-dire ne possèdant que des 1 dans son écriture décimale. Implémenter une procédure `chiffre_un(n)` qui retourne — dans les cas favorables — le premier entier constitué uniquement de 1 multiple de n ainsi que le nombre de 1 dans son écriture.

Par exemple, comme d'une part $111 = 37 \times 3$ et d'autre part 1 et 11 ne sont pas multiple de 37, `chiffre_un(37)` doit renvoyer `(111,3)`.




```
You can do a lot better than 1%
if you start keeping track of the patterns in what numbers people pick.
```

Partie VI

Nombres stables

On souhaite savoir comment de fois on écrit le chiffre 1 lorsque l'on écrit tous les entiers naturels de 1 à 2014.

1. Écrire une fonction `nb_un(n)` qui prend comme entrée un entier naturel n et qui retourne le nombre de fois que le chiffre 1 apparaît dans n. (À titre d'exemple, on a `nb_un(51218)` doit renvoyer 2.)
2. Combien de fois utilise-t-on le chiffre 1 pour coucher sur le papier tous les entiers compris entre 1 et 2014 ?

On dit que p est un nombre *stable* lorsque le nombre de 1 nécessaire pour écrire tous les entiers de 1 à p, soit exactement p. Par exemple il est facile de voir que 0 et 1 sont des nombres stables. On peut se demander s'il en existe d'autre.

3. Construire une fonction `liste_stable(n)` prenant comme entrée n et retournant la liste de nombres stables inférieur ou égaux à n

Partie VII

Nombres autoréférents

Soit n un nombre entier naturel. On lui associe le nombre entier $f(n)$ que l'on construit suivant la règle suivante :

• On note ℓ le nombre de chiffres de n dans son écriture décimale. Si l'un des chiffres de n n'est pas dans $\{0, 1, \dots, \ell - 1\}$ alors $f(n) = 0$.

Dans tous les autres cas,

◇ Le premier chiffre de $f(n)$ est le nombre de 0 apparaissant dans n.

◇ Le second chiffre de $f(n)$ est le nombre de 1 apparaissant dans n.

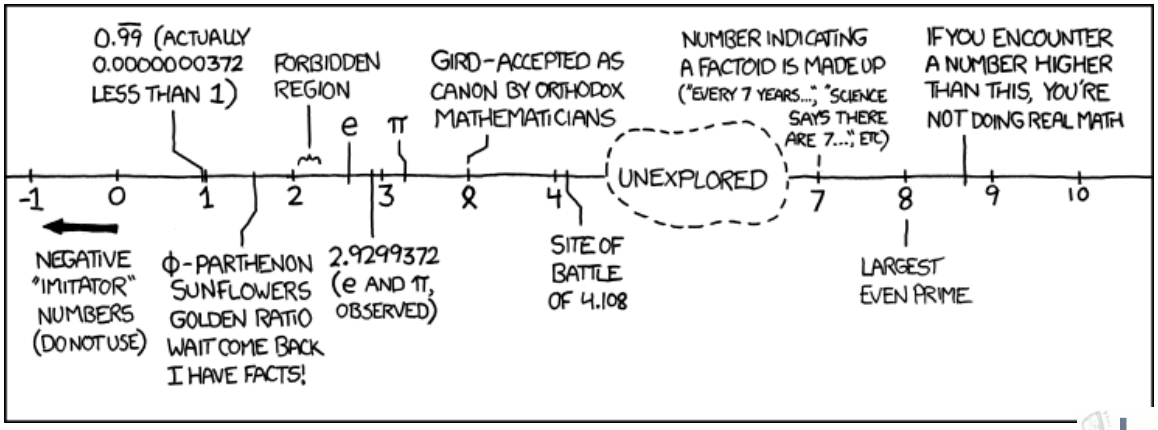
◇ Le troisième chiffre de $f(n)$ est le nombre de 2 apparaissant dans n et ainsi de suite jusqu'à la longueur ℓ du nombre $2 \leqslant \ell \leqslant 10$.

Par exemple, on a $f(2520) = 0$ puisque 2520 possède quatre chiffres et $5 \notin \{0, 1, 2, 3\}$. On calcule aussi

$$f(1112) = 0310 = 310 \qquad \text{et} \qquad f(222) = 003 = 3.$$

On dit qu'un nombre est *autoréférent* lorsque $f(n) = n$.

1. Implémenter la fonction f décrite ci-dessus.
2. Trouver tous les nombres autoréférents de $[\![0, 10^7]\!]$



The Wikipedia page "List of Numbers" opens with
"This list is incomplete; you can help by expanding it."

Partie VIII

Nombres premiers

1 Décomposition

Soit n un entier naturel non nul. On dit que n est un nombre *premier* lorsqu'il admet exactement deux diviseurs distincts entiers et positifs (qui sont alors 1 et lui-même). Cette définition exclut 1, qui n'a qu'un seul diviseur entier positif.

1. Implémenter une fonction `is_prime(n)` qui prend en entrée un entier naturel n plus grand que 1 et qui retourne `True` lorsque n est un nombre premier et `False` sinon.
2. Implémenter une fonction `ith_prime` qui prend en entrée un entier naturel n non nul et qui retourne le n^{e} nombre premier. En particulier `ith_prime(1)` renvoie 2, `ith_prime(2)` renvoie 3, `ith_prime(3)` renvoie 5, `ith_prime(4)` renvoie 7 et `ith_prime(5)` renvoie 11.
3. Construire une fonction `decomposition(n)` qui prend comme entrée un entier naturel $n \geqslant 2$ qui retourne la liste des couples formé des diviseurs premiers de n avec leur ordre de multiplicité. Par exemple, `decomposition(36)` doit renvoyer `[[2,2],[3,2]]` car $36 = 4 \times 9 = 2^2 \times 3^2$
4. Construire une fonction `voir_decomposition(n)` qui permet de mieux visualiser la décomposition de n en produit de nombres premiers. En particulier, `voir_decomposition(360)` devra produire `2^3 * 3^2 * 5^1`

2 Crible d'Erathosthène

Implémentez la fonction `crible(n)` qui suit la procédure du crible d'Erothosthène dont le fonctionnement est expliqué ci-après. L'algorithme procède par élimination : il s'agit de supprimer d'une table des entiers de 2 à n tous les multiples d'un entier. En supprimant tous les multiples, à la fin il ne restera que les entiers qui ne sont multiples d'aucun entier, et qui sont donc les nombres premiers. De manière détaillée:

- On commence par rayer les multiples de 2, puis à chaque fois on raye les multiples du plus petit entier restant.
- On peut s'arrêter lorsque le carré du plus petit entier restant est supérieur au plus grand entier restant, car dans ce cas, tous les non-premiers ont déjà été rayés précédemment.
- À la fin du processus, tous les entiers qui n'ont pas été rayés sont les nombres premiers inférieurs à N.

3 Nombres squarefree

Un entier naturel non nul est *squarefree* ou *quadratfrei* (sans facteur carré) lorsqu'il n'est divisible par aucun carré parfait autre que 1.

1. Construire une fonction booléenne `is_squarefree(n)` qui retourne `True` si n est squarefree et `False` sinon.
2. Construire une procédure `squarefree(n)` de la variable n qui retourne la liste de tous les entiers squarefree inférieurs ou égaux à n.

Partie IX

Nombres aux mêmes chiffres (ProjectEuler n°52)

On constate facilement que le nombre 125 874 et son double 251 748 contiennent exactement les mêmes chiffres dans leur écriture décimale mais dans un ordre différent. Trouvez le plus petit entier positif n tel que $2n$, $3n$, $4n$, $5n$ et $6n$ contiennent tous les mêmes chiffres.

Partie X

Nombres de Kaprekar

Dattatreya Ramachandra Kaprekar (1905 - 1986) né à Dahanu près de Bombay est un mathématicien indien connu pour ses recherches sur les nombres. On lui doit la notion de nombre de Kaprekar ainsi que l'algorithme de Kaprekar. Boudé par ses contemporains, ses travaux seraient passés inaperçus s'il n'avait pas été relayés par Martin Gardner, spécialiste de mathématiques récréatives. Passionné par les nombres, il dira : « Un alcoolique souhaite continuer à boire pour retrouver un état de plaisir. Il en est de même pour moi concernant les nombres. »

Nous allons étudier l'algorithme de Kaprekar découvert en 1949. On considère un entier naturel à quatre chiffres, par exemple $n = 5278$. On associé à cet entier le nombre $n' = 8752$ obtenu en reclassant les chiffres formant 5278 dans l'ordre décroissant et $n'' = 2578$ en reclassant ces même chiffres dans l'ordre croissant.

La différence $n' - n''$ entre ces reclassements : $8752 - 2578 = 6174$ est appelé K-itéré de 5278. Nous allons remarquer ce qu'il se passe lorsque l'on calcule les K-itérés d'un entier à quatre chiffre donné. Par exemple, en partant de 1826, on obtient

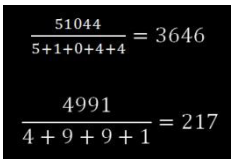
$$1826 \ \rightarrow \ 7353 \ \rightarrow \ 4176 \ \rightarrow \ 6174$$

puis le procédé s'arrête.

1. Écrire une fonction `chiffre(n)` qui prend comme entrée un entier naturel n et qui retourne la liste des chiffres dans l'écriture décimale de n. Par exemple `chiffre(24536)` doit renvoyer `[2,4,5,3,6]`.
2. Écrire une fonction `decroissant(n)` qui prend comme entrée en entier naturel n et qui retourne nombre entier obtenu en reclassant les chiffres formant n dans l'ordre décroissant. Par exemple `decroissant(24536)` doit renvoyer `65432`.
3. Écrire une fonction `croissant(n)` qui prend comme entrée en entier naturel n et qui retourne nombre entier obtenu en reclassant les chiffres formant n dans l'ordre croissant. Par exemple `croissant(24536)` doit renvoyer `23456`.
4. Écrire une fonction `kaprekar(n)` qui prend comme entrée un entier naturel n et qui retourne $n' - n''$ en utilisant les notations définies ci-dessus. Par exemple `kaprekar(1826)` doit renvoyer `7353` car $8621 - 1268 = 7353$
5. Montrer que si on définit la fonction `kaprekar(n)` sur les nombres entiers à quatre chiffres, celle-ci est à valeurs dans l'ensemble des entiers à quatre chiffres auxquels on ajoute 0 et 999. On notera $\Omega = [\![1000, 9999]\!] \cup \{0, 999\}$.
6. Trouver les points fixes de la fonction `kaprekar(n)` contenus dans l'ensemble Ω.
7. Écrire une fonction `kaprekar_it(n,i)` qui prend comme entrée les entiers naturels n et i qui retourne le i^{e} itéré de n par la fonction `kaprekar(n)`.
8. Montrer que si n est un entier de $[\![1000, 9999]\!]$ alors le 7^{e} itéré de n est dans $\{0, 6174\}$.
9. Écrire une fonction `chaine(n)` qui prend comme entrée un entier naturel n et qui retourne la liste des itérés de n jusqu'à ce que l'on atteigne 0 ou 6174. Par exemple `chaine(1826)` doit renvoyer `[1826,7353,4176,6174]` alors que `chaine(1211)` doit renvoyer `[1211,999,0]`.
10. Donner un exemple de chaine à 8 éléments.

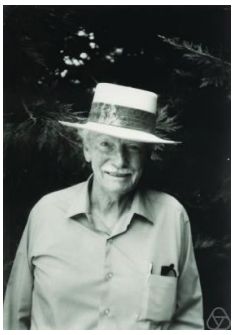
Partie XI

Nombres de Niven



Nous allons dans cet exercice implémenter une procédure permettant de déterminer les nombres Harshad ou nombres de Niven.

Le terme Niven est un hommage au mathématicien Ivan Niven qui a publié un article et présenté une conférence en théorie des nombres sur leur sujet en 1977.



Ivan Morton Niven (né le 25 octobre 1915 à Vancouver et mort le 9 mai 1999 à Eugene, dans l'Oregon) est un mathématicien américain et canadien spécialiste de la théorie des nombres.

Le nom de Harshad leur a été donné par le mathématicien Dattatreya Ramachandra Kaprekar et signifie en sanskrit grande joie.

Par définition, un nombre Harshad, ou nombre de Niven, est un entier qui est divisible par la somme de ses chiffres dans son écriture décimale.

1. Écrire une procédure `somme` qui à un entier n associe la somme de ses chiffres dans son écriture décimale.
2. Écrire une procédure `Niven` qui prend comme entrée en entier n et qui retourne `True` si n est de Niven et `False` dans le cas contraire.

La mathématicienne anglaise Helen G. Grundman a démontré qu'il n'existe pas 21 entiers consécutifs qui sont tous des nombres de Niven. Elle trouva aussi la plus petite suite de 20 entiers consécutifs qui sont tous des nombres Harshad. Ils dépassent tous $10^{44363342786}$.

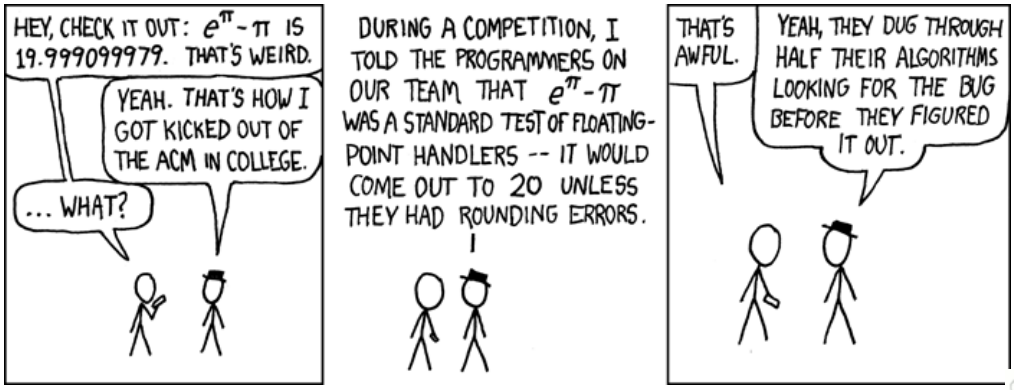
3. Déterminer le listes des nombres de Niven plus petit que 100.
4. Trouver une liste de 7 nombres de Niven consécutifs ayant tous plus de deux chiffres dans leur écriture décimale.
5. Trouver la plus petite liste de 17 nombres consécutifs qui ne sont pas de Niven.

Partie XII

Nombres ploutons

Le célèbre mathématicien indien Srinivasa Ramanujan appelait nombre hautement composé (ou nombre plouton) un entier qui a strictement plus de diviseurs que tous ses prédecesseurs.

1. Construire une procédure `nb_div(n)` qui prend comme entrée un entier naturel n non nul et qui retourne de nombre de diviseur de n.
2. Faire une procédure `ramanujan(n)` qui prend comme entrée un entier naturel $n \geqslant 2$ et qui affiche la liste des entiers de Ramanujan inférieurs à n.




```
Also, I hear the 4th root of (9^2 + 19^2/22) is pi.
```

Partie XIII

Somme de nombres

1. Proposer *des* procédures `somme_chiffres(n)` prenant comme entrée un entier naturel n et qui retourne la somme des chiffres dans l'écriture décimale de n.
2. Nous allons maintenant résoudre les problèmes 16 et 20 du project Euler:
   a) Trouvez la somme des chiffres de l'écriture décimale de 100!.
   b) Trouvez la somme des chiffres de l'écriture décimale de 2^{1000}.

Partie XIV

Nombres amicaux

On va dans un premier temps construire plusieurs procédures qui retournent le nombre de diviseurs strict (*ie* strictement inférieur à n) d'un entier naturel n non nul donné.

1. Implémenter une méthode naïve consistant à tester tous les entiers jusqu'à n pour compter les diviseurs de n.
2. En remarquant que les diviseurs « marchent par deux » améliorer la procédure ci-dessus. On donne le Pseudo-Code suivant que vous aurez à comprendre sur des exemples et à implémenter. On le testera en particulier pour $n = 12$ et $n = 36$.

```
Entrées : n (un entier naturel non nul)
c ← 0                          # initialisation du compteur
fin ← n                        # La mystérieuse variable fin
k ← 1                          # le premier candidat à être diviseur de n
tant que k < fin faire
    si n%k == 0 alors
        fin ← n//k
        si k == fin alors
            On incrémente c de 1
        sinon
            On incrémente c de 2
        fin
        On incrémente k de 1
    fin
fin
retourner c
```

Algorithme 1: Nombre de diviseurs distincts de n

3. Modifier la procédure ci-dessus afin de retourner le nombre de diviseurs stricts de n c'est-à-dire le nombre de diviseurs de n distincts de n.
4. Modifier la procédure ci-dessus afin de retourner la liste des diviseurs de n.

Si l'on appelle $d(n)$ la somme des diviseurs stricts de n alors un nombre n est dit amical si $d(d(n)) = n$, c'est-à-dire si le nombre résultant de la somme des diviseurs stricts de n admet n lui-même comme somme de ses diviseurs stricts. On remarque que si n est un nombre amical alors $d(n)$ l'est aussi automatiquement (d'où le terme « amical »: les deux nombres sont amis entre eux).

Par exemple 220 et 284 constituent la plus petite paire de nombres amicaux. Les diviseurs stricts de 220 sont $1, 2, 4, 5, 10, 11, 20, 22, 44, 55$ et 110 dont la somme fait 284 et les diviseurs stricts de 284 sont $1, 2, 4, 71$ et 142 dont la somme fait 220.

5. Trouver la somme de tous les nombres amicaux en-dessous de 10 000 (Project Euler n°21).

Partie XV

Palindromes

Un nombre palindrome est un nombre dont l'écriture est la même quel que soit le sens de lecture. Par exemple 1234321 est un palindrome.

1. Écrire une fonction `palindromes_divisibles_par(n)` qui renvoie la liste de tous les palindromes inférieurs à un million divisibles par l'entier n. On pourra de plus afficher à l'écran le nombre de palindromes de k chiffres divisibles par n pour $k \in [\![1 ; 5]\!]$.
2. Regarder ce que renvoie `palindromes_divisibles_par(9)`. La tendance affichée continue-t-elle avec les nombres à 6 ou 7 chiffres ?


```
StackSort connects to StackOverflow, searches for 'sort a list',
and downloads and runs code snippets until the list is sorted.
```