Rappels

On rappelle que `a % b` calcule le reste de la division euclidienne de a par b et que `a // b` donne son quotient.

Les listes

N'hésitez pas à consulter les énoncé des TD 3 et 4 pour les différentes opérations sur les listes. En particulier la construction suivante nous sera très utile !

```
def ....
    L = []
    for .... ou while ....
        ....
        L.append(un_element)
    return # mettre ici la valeur le retour demandée.
```

On fait tourner mais ça reste premier

Le nombre 113 est remarquable dans le sens suivant : il est premier et les 2 autres nombres obtenus par permutation cyclique de ses chiffres le sont aussi : 311 et 131 sont premiers. De plus, 113 est le plus petit nombre de 3 chiffres possédant cette propriété.

On se propose de déterminer, s'il existe, le plus petit nombre de 6 chiffres qui possède cette propriété.

Il est interdit d'utiliser des fonctions Python répondant immédiatement aux questions !

1. Expliquer pourquoi la fonction **est_premier** retourne bien un booléen qui indique si le nombre passé en paramètre est premier ou non.
   Remarque : on prouve mathématiquement qu'un nombre n est premier ssi pour tout entier $k > 1$ tel que $k^2 <= n$, k n'est pas un diviseur de n.
2. Écrire une fonction `nombre_liste` prenant une liste d'entiers entre 0 et 9 et retournant le nombre dont ce sont les chiffres. Par convention, on écrit les chiffres dans la liste dans l'ordre inverse l'ordre de lecture.

   Exemple : `nombre_liste([5, 7, 8])  :  875`

   Question : à quel notion mathématique correspond l'indice dans la liste des chiffres ? Dans notre exemple (et en notant $\mathrm{L} = [5, 7, 8]$), quel est le rapport entre 0, L[0], 1 L[1], 2, L[2] et le nombre 875
3. Ecrire une fonction `liste_chiffres` prenant en paramètre un entier naturel non nul n (ne pas vérifier dans le code que $n \in \mathbb{N}^*$) et renvoyant la liste des chiffres de l'écriture de n en base 10. Le chiffre des unités doit être en première position.
   On pourra s'inspirer du cours sur la décomposition en binaire !

   Exemple : `liste_chiffres(578)  :  [8, 7, 5]`

   Effectuer les opérations correspondant à un appel à `liste_chiffre` (sur une feuille, ou dans la console python) avec n=578.
4. Écrire une fonction `decale_liste` prenant en paramètre une liste non vide et renvoyant la liste obtenue en décalant les éléments de la liste de manière circulaire, le dernier élément prenant la place du premier.

   Exemple : `decale_liste([1, 7, 8, 9])  :  [9, 1, 7, 8]`
5. Écrire une fonction `est_solution` prenant en paramètre un entier naturel n (comportant un nombre a priori quelconque de chiffres en écriture décimale) et renvoyant `True` ou `False` suivant que n vérifie les conditions du problème ou pas.

   Exemples : `est_solution(113)  :  True`   `est_solution(2013)  :  False`

6. Écrire une fonction `cherche_solution` prenant en paramètre un entier naturel N et renvoyant le plus petit nombre comportant N chiffres qui soit solution du problème s'il en existe un. S'il n'existe pas de solution, la fonction ne devra rien renvoyer.

   Exemple : `cherche_solution(3)  :  113`

   Première étape : quels sont les nombres à N chiffres ? Tester votre réponse avec $N = 3, 4$.
7. Répondre au problème posé en préambule.

Exercices

Un ensemble d'entiers est représenté par une liste strictement croissante.

Exercice 1
Créer une fonction **est__str__croissante(L)** qui vérifie que la liste L est une liste d'entier strictement croissante et renvoi un booléen.

Exercice 2
Ecrire une fonction `intersection` prenant en paramètre deux listes d'entiers (supposées strictement croissantes, on ne le vérifiera pas dans le code) et renvoyant la liste intersection.

Exercice 3
Faire de même avec `reunion`

Exercice 4
Comment tester si un entier donné est dans un ensemble ?
Comment ajouter un nombre à un ensemble d'entier ? (attention, un ensemble ne comporte pas deux fois le même élément). (indication : `L.insert`)

Mode expert

Exercice 5
Etant donné un ensemble L (représenté par une liste strictement croissante, de longueur n), calculer la liste de tous ses sous-ensembles. Il y en a exactement 2^n dont $\binom{n}{k}$ possédant k éléments pour chaque $k \in [\![0, n]\!]$. Pour ce faire, on pourra encoder un sous-ensemble par une représentation binaire de n bits (chaque bit représentant une appartenance ou non à la partie en construction).

Exercice 6
Reprendre l'exercice 4, mais en trouvant des algorithme dont la complexité (mesurée en nombre de tests) est au maximum proportionnelle à $\ln(n)$ où n est la longueur de la liste.