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 l' énoncé des TP précédents 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)  # ajoute un élément à la fin de L
    return # mettre ici la valeur le retour demandée.
```

Échauffement

Décrire rapidement la valeur de retour de la fonction suivante. On pourra éventuellement faire un tableau d'évolution de la valeur des différentes variables locales (celles qui sont définies dans la fonction).
En particulier, quelles sont les valeurs successives contenues dans la variable `p` ?

```
def f(n):
    L = [(-1)**k for k in range(0, n + 1)]
    p = 1
    s = L[0] / p
    for i in range(1, n + 1):
        p = p * i
        s = s + L[i] / p
    return s
```

Cette fonction est présente dans le fichier **tp8.py**. Modifier légèrement le code fourni pour voir apparaître à l'écran les valeurs successives de p et s lors du calcul de `f(10)` (ou de tout autre entier de votre choix).

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** (présente dans le fichier `tp8.py`) retourne bien un booléen. Ce booléen indique si le nombre passé en paramètre est premier ou non.
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`

   Indication : à quelle notion mathématique correspond l'indice dans la liste des chiffres ? Dans notre exemple (et en notant 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. Intermède. Rappeler, puis vérifier le résultat dans la console, la sortie attendue après chacune des commandes

```
L = [1,3,5,7,9,11]
```

```
L[-1]
```

```
L[2:4]
```

```
L[:-1]
```

5. 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]`
6. 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`
7. 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 : `return None`

   Exemple : `cherche_solution(3)  :  113`

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

Exercices

Un ensemble d'entiers est représenté par une liste strictement croissante. Par exemple l'ensemble des entiers naturels pairs inférieurs à 11 est représenté par la liste `[0, 2, 4, 6, 8, 10]`

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 5, 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.