Partiel d'Informatique-Algorithmique
Mardi 23 octobre 2018 – Durée : 1h50

Documents papier et informatique autorisés – Accès internet et téléphones portables interdits.

Important

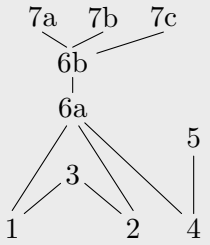
A. Créez votre répertoire de partiel `E2_Nom_Prenom` sur le « bureau » de l'ordinateur. À la fin, il devra contenir seulement deux fichiers : `numchiffre.py` et `kaprekar.py`. Lorsque vous avez terminé ou que le temps imparti est écoulé, vous devrez remettre votre répertoire de partiel à l'enseignant·e ou imprimer les fichiers et les rendre sous format papier.

B. Il est nécessaire de remplir les en-têtes des fonctions et des modules, de **tester vos fonctions**, et de commenter lorsque cela vous semble nécessaire.

C. N'oubliez pas que si vous avez un doute sur ce que produit une expression, vous pouvez la tester dans l'interpréteur (le shell) Python ou en afficher la documentation en tapant `help`.

D. **Sauvegardez régulièrement votre travail.**

E. Les questions dépendent les unes des autres :



Par exemple il est indispensable de traiter les questions 1 et 2 et 4 avant d'aborder la question 6a. Si vous n'arrivez pas à écrire une fonction, simulez son fonctionnement en lui faisant retourner une valeur du même type que celui qui est demandé dans l'énoncé, et passez à la suite.

Introduction

Un nombre de Kaprekar[1] est un entier naturel qui, lorsqu'il est élevé au carré, peut être séparé en une partie gauche et une partie droite dont la somme donne le nombre initial. Par exemple 55 est un nombre de Kaprekar car $55^2 = 3025$ et $55 = 30 + 25$, et 12 n'en est pas un car $12^2 = 144$ mais ni $1 + 44$, ni $14 + 4$, ni 144 ne sont égaux à 12. Il n'y a pas besoin que les deux parties aient le même nombre de chiffres, ni qu'elles commencent par un chiffre non nul, ainsi par exemple 4879 est un nombre de Kaprekar puisque $4879^2 = 23804641$ et $238 + 04641 = 4879$.

Travail à rendre

Module `numchiffre.py`

Créer un module `numchiffre.py` qui contiendra les en-têtes habituels, les définitions des fonctions ci-dessous (questions 1 et 2) et les tests (question 3).

1. (3 points) Créer une fonction `liste_chiffres` avec un paramètre entier, qui renvoie la liste de ses chiffres en base 10. Par exemple l'appel `liste_chiffres(729)` devrait renvoyer la liste `[7,2,9]`. *Les fonctions de transtypage, en particulier `str`, ainsi que la fonction `reverse` sont autorisées, mais pas obligatoires, pour cette question ; on rappelle que si `chaine_chiffres` est une chaîne de caractères ne comprenant que des chiffres, alors `[int(c) for c in chaine_chiffres]` produit une liste d'entiers.*

2. (3 points) Créer une fonction `forme_nombre` qui prend en argument une liste de chiffres (entiers entre 0 et 9) et renvoie le nombre formé sur ces chiffres. Par exemple un appel sur la liste `[0,2,1,7]` devrait renvoyer l'entier `217`. *Vous pouvez vous aider de l'une des deux écritures suivantes (de votre choix)*

$$217 = \mathbf{7} \times 10^0 + \mathbf{1} \times 10^1 + \mathbf{2} \times 10^2 + \mathbf{0} \times 10^3 \quad (1)$$
$$= \mathbf{7} + 10\,(\mathbf{1} + 10\,(\mathbf{2} + 10 \cdot \mathbf{0}))\,. \quad (2)$$

On pourra utiliser une boucle `for`.

1. Dattatreya Ramachandra Kaprekar, 1905 -1986.

3. (2 points) Importer la fonction `randrange` du module `random` à l'endroit adapté dans `numchiffre.py`. Dans le corps du module `numchiffre.py` ajouter les instructions :

```
if __name__ == "__main__" :
    for j in range(98,102):
        print (j,forme_nombre(liste_chiffres(j)))
    liste_test = []
    for k in range(5):
        liste_test.append(randrange(10))
    print (liste _test, forme_nombre(liste_test))
```

Executer `numchiffre.py`, recopier le résultat d'execution à la fin et commenter : le résultat est-il conforme à ce qui est attendu ?

Programme principal

Créer un fichier `kaprekar.py`, importer le module `numchiffre.py`.

4. (3 points) Créer une fonction `decoupe` qui prend en argument deux paramètres : une liste `p_liste` et un entier positif `p_dec` et qui renvoie deux listes, l'une contenant les éléments de `p_liste` de 0 à `p_dec` (inclus), l'autre contenant les éléments de `p_liste` en positions de `p_dec+1` à la fin. *On pourra renvoyer le résultat sous forme de tuple ou bien de liste, selon votre préférence.* Voici quelques exemples :
   — `decoupe([3,4,3,2,1,2,1,0],4)` devrait renvoyer le tuple `([3,4,3,2,1],[2,1,0])` ou bien la liste `[[3,4,3,2,1],[2,1,0]]`.
   — `decoupe([5,0,9,2,8,4,1,5,7],0)` devrait renvoyer le tuple `([5],[0,9,2,8,4,1,5,7])` ou bien la liste `[[5],[0,9,2,8,4,1,5,7]]`.
5. (1 ½ points) Ecrire des instructions permettant de tester la fonction `decoupe`. Recopier vos résultats d'execution à la fin du programme.
6. (3 points) (a) Ecrire une fonction `somme_parties` qui prend en argument un entier naturel et renvoie la liste des sommes des parties de son écriture décimale. Par exemple `somme_parties(3025)` devrait renvoyer `[28,55,307,3025]` car $3 + 025 = 28$ et $30 + 25 = 55$ et $302 + 5 = 307$ et $3025 = 3025$. *On appelera les fonctions précédemment définies.*

   (b) Ecrire une fonction `est_kaprekar` qui prend en argument un entier naturel `p_entier` et renvoie un booléen dont la valeur est `True` si `p_entier` est un nombre de Kaprekar et `False` sinon. *Il est recommandé de relire la définition et les exemples avant d'écrire cette fonction.* L'opération `in` est autorisée pour rechercher dans la liste renvoyée par `somme_partie`.
7. (4 ½ points) (a) Dans le corps du programme principal, écrire les instructions nécessaires pour demander à l'utilisateur·trice de saisir un entier naturel (refuser tant qu'il n'est pas positif) et répondre s'il s'agit d'un nombre de Kaprekar ou non. Recopier les résultats d'execution à la fin (on pourra essayer : `22,-4,9,297,7777`).

   (b) Donner tous les nombres de Kaprekar entre 1 et 1000 (on ne demande pas les décompositions correspondantes). *Vous devez en trouver* 11.

   (c) Il existe un unique nombre de Kaprekar strictement compris entre 82600 et 83000. Quel est-il ? Vous pourrez utiliser une boucle `for` ou `while` en commentant votre choix.

Bonus (seulement si tout le reste est parfait)

Il n'est pas nécessaire de traiter ces questions pour avoir une excellente note.

8. (½ point) Réécrire les fonctions du module `numchiffre.py` de manière récursive.
9. (½ point) Donner une liste des petits nombres [2] de Kaprekar en base 2.

2. Douglas Ianucci a démontré en 2000 que cette liste contient tous les nombres pairs parfaits (égaux à la somme de leurs diviseurs différents d'eux-même). Sa preuve repose sur un théorème d'Euler (1747), qui dit que tous ces nombres sont produits par une construction dûe à Euclide ($\sim$ - 300) : ils sont de la forme $2^{p-1}(2^p - 1)$ où p est un nombre premier tel que $2^p - 1$ est premier. On ne sait pas s'il existe une infinité de nombres parfaits, ni s'il en existe un qui soit impair.