L'intégramme de Lewis Caroll

L'énigme des 5 maisons.

Les intégrammes sont des petits jeux de logique semblables à des casse-têtes abstraits. Partant d'un certain nombres d'indices sur des éléments, le jeu consiste à déduire des relations non triviales entre ceux-ci, de manière à résoudre, au final, une énigme qui ne semblait pas corrélée aux indices initiaux.

L'un des intégrammes le plus célèbre est celui des 5 maisons, que l'on attribue tour à tour à Lewis Caroll ou à Einstein, bien qu'aucune preuve de la paternité de l'un ou de l'autre n'ait jamais été apportée. Lewis Caroll jouant aussi bien avec les mots [1] qu'avec les symboles mathématiques [2], on l'imagine sans peine poser l'énigme suivante :

Cinq voisins de nationalités et de professions différentes habitent les cinq premières maisons d'une même rue dont les façades sont singulières. Adepte d'une boisson particulière, chacun d'eux partage sa vie avec son animal de la façon suivante :

1. L'Anglais habite la maison rouge.
2. L'Espagnol adore son chien.
3. L'Islandais est ingénieur.
4. La maison verte sent bon le café.
5. La maison verte est située immédiatement à gauche de la blanche.
6. Le sculpteur possède un âne.
7. Le diplomate habite la maison jaune.
8. Le Norvégien habite la première maison à gauche.
9. Le médecin habite la maison voisine de celle où demeure le propriétaire du renard.
10. Le diplomate voit un cheval dans le jardin voisin tous les matins en ouvrant ses volets.
11. La maison du milieu reçoit une livraison de lait tous les mardis.
12. Le Slovène boit du thé.
13. Le violoniste presse ses oranges à la main pour son jus quotidien.
14. Le Norvégien jalouse le joli bleu de la maison voisine.

Qui élève un zèbre ?

1. Vous connaissez bien sûr *Les Aventures d'Alice aux pays des Merveilles...*
2. ...mais avez-vous déjà entendu parler de *l'algorithme de condensation de Dodgson* ? Selon la nature de ses activités, Lewis Caroll jonglait entre deux identités qu'il dissociait entièrement. Mathématicien, il signait alors de son nom de naissance : Charles Lutwidge Dodgson.

Avec un peu d'astuce, l'énigme se laisse résoudre à la main. Pourtant, si la patience vous manque, il est intéressant de savoir que l'on peut reléguer ce type de tâche à son ordinateur. Avant de faire travailler Python, nous allons "relaxer" les hypothèses pour nous permettre malgré tout d'obtenir un résultat sans attendre plusieurs jours. Pour les curieux, l'explication viendra plus loin : retenez qu'autrement, Python serait vaillant mais lent. Le problème simplifié sur lequel je vous propose de vous pencher est le suivant :

Enigme des 5 maisons pour Python. Cinq voisins de nationalités et de professions différentes habitent les cinq premières maisons d'une même rue. Chacun d'eux partage sa vie avec son animal de la façon suivante :

1. L'Anglais est sculpteur ou médecin, nul ne sait.
2. L'Espagnol adore son chien.
3. L'Islandais est ingénieur.
4. Le sculpteur possède un âne.
5. Le Norvégien habite la première maison à gauche.
6. Le médecin habite la maison voisine de celle où demeure le propriétaire du renard.
7. Le diplomate voit un cheval dans le jardin voisin tous les matins en ouvrant ses volets.
8. Le Slovène ne comprend pas son voisin Islandais.
9. Le violoniste ne parle pas norvégien. Tant mieux, son cheval non plus.
10. Le chien a peur du renard et a obligé son maître à déménager pour ne plus être côte à côte.
11. L'Islandais est rassuré d'avoir un voisin médecin depuis son cancer de l'orteil, et le Norvégien partage son avis pour se faire prescrire des ampoules de vitamine D plus facilement.

Qui élève un zèbre ?

Résolution de l'énigme en Python.

Les questions qui suivent sont volontairement moins guidées qu'à l'habitude. Pour certaines, des pistes de résolution se trouvent en fin de document. Je vous conseille pourtant de réfléchir au mieux avant d'aller lire ces indices – et si vous pouvez vous en passez, c'est encore mieux !

Question 1 : Structures de données. Votre algorithme va manipuler plusieurs ensembles de données : les nationalités, les professions, les animaux, – et, éventuellement, les emplacements, bien que l'on puisse faire sans. Il est donc naturel de commencer par déclarer ces 3 groupes de mots, chacun ayant 5 éléments. Parmi les listes, les n-uplets, les ensembles et les dictionnaires, quelle est la structure de données qui vous semble la plus appropriée ? Pourquoi ? Déclarez vos 3 premiers ensembles **nati, prof** et **anim** en

conséquence.

Avant de répondre à la question de savoir qui élève un zèbre, vous allez devoir créér et tester différentes configurations, c'est-à-dire différentes combinaisons d'éléments à l'intérieur d'une même maison. Par exemple, on souhaite pouvoir faire comprendre à Python que le sculpteur anglais a un âne (c'est un exemple, pour le moment je suis comme vous : je n'en sais rien). Quel type de données vous semble approprié ? Déclarez un alias **maison** pour alléger les notations.

Question 2 : Squelette de l'algorithme. Sans chercher une manière intelligente de procéder, proposez une méthode pour résoudre le problème, en décomposant celui-ci en plusieurs sous-étapes. En particulier, je vous déconseille d'essayer de calquer la méthode astucieuse que vous pourriez avoir envie d'appliquer si vous cherchiez à résoudre l'énigme "à la main". Un algorithme un peu brutal nous suffira *tant que cela fonctionne.* Donnez les signatures des fonctions que vous souhaitez créer.

Question 3 : Liste des permutations d'une liste. Ecrire la fonction **permutation** dont la spécification est la suivante :

```
def permutation(L):
    """list[alpha]->list[list[alpha]]
    Hypothese : len(L)!=0
    retourne la liste composée de toutes les listes possibles crées
    à partir des éléments de L et dont les éléments ont été permutés.
    La liste de retour a donc pour longueur len(L)!"""
```

Par exemple :

```
>>>permutation([1,2,3])
[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Question 4 : Traduction des contraintes. Ecrivez la fonction **verification** dont la spécification est la suivante :

```
def verification(config):
    """list[maison]->bool
    Hypothese : config est une liste de maison utilisant une seule et unique
    fois chaque critère (par exemple, il ne peut y avoir 2 ânes dans la rue).
    Retourne True si et seulement si config vérifie toutes les contraintes
    de l'enigme.
    """
```

Question 5 : Recherche exhaustive. Ecrivez la fonction **recherche** dont la spécification est :

```
def recherche():
    """->list[maison] + str
    Retourne une configuration qui satisfait toutes les conditions, si elle
    existe, et un message d'erreur sinon.
    """
```

Question 6 : Solution finale. Ecrivez la fonction **solution** sans paramètre qui répond à la question posée.... et lancez votre fonction ! Un mot à dire sur le temps d'exécution ?

Indices

- **Question 2** : Partez du principe que vous pouvez tester toutes les configurations possibles. A partir de là, il s'agit :
  1. de les créer.
  2. de les tester une par une.

  Lire les intitulés des questions suivantes peut aussi vous aider...
- **Question 3** : Passez par une fonction intermédiaire de spécification :

  ```
  def permutation_entiers(n):
      """int->list[list[int]]
      Hypothese : n >=1
      Retourne la liste des n! permutations possibles des entiers
      compris entre 0 et n-1. Une permutation est representee
      elle-meme par une liste d'entiers, celle des images."""
  ```

  Que vous pourrez définir par récurrence, c'est-à-dire qu'elle peut "s'appeler elle-même". Pour construire cette dernière vous pouvez vous aider de la fonction (à implémenter) :

  ```
  def modifierliste(L,i,n):
      """list[int]*int*int->list[int]
      Hypothese : i apparait exactement une fois dans la liste L,
      qui est au moins de longueur 1
      Retourne la liste L ou l'on a remplace l'entier i (qui apparait)
      par l'entier n"""
  ```

  Par exemple,

  ```
  >>>modifierliste([1,2,3],3,5)
  [1,2,5]
  >>>permutation_entiers(3)
  [[0, 2, 1], [0, 1, 2], [1, 0, 2], [1, 2, 0], [2, 0, 1], [2, 1, 0]]
  ```
- **Question 4** : Séparez les conditions spatiales des autres et traitez les une fois les conditions plus faciles étudiées.

– **Question 5** : Cette question est plutôt simple si vous vous servez correctement des (corrigés des) fonctions précédentes.