ISEN ALL IS DIGITAL! BREST | yncréa

Jean-Marie Guyader
jean-marie.guyader@isen-ouest.yncrea.fr
Bureau 330

Année 2020-2021

Projet CENT 2 – *Le compte est bon !*

Objectifs

Le but de cette séance est de réaliser un algorithme de résolution automatique des problèmes du jeu *Le compte est bon*, diffusé dans l'émission *Des chiffres et des lettres*.

1. Règles du jeu

Six nombres entiers positifs, compris entre 1 et 100 et tirés aléatoirement, doivent être utilisés pour trouver un septième nombre, appelé *objectif*, ce dernier étant tiré aléatoirement entre 101 et 999. Quatre opérations mathématiques peuvent être utilisées à cet effet : l'addition, la soustraction, la multiplication et la division. Chacun des six nombres d'origine ne peut être utilisé qu'une seule fois. Chacun des résultats intermédiaires ne peut également être utilisé qu'une fois. Il est interdit d'obtenir comme résultat intermédiaire ou final un nombre qui soit décimal, fractionnaire, ou négatif.

2. Exemple de compte

La Figure 1 présente un exemple de compte à résoudre. Sur fond bleu, le nombre 902 de la première ligne est l'objectif à atteindre. Les nombres sur fond violet de la deuxième ligne doivent servir comme base pour calculer l'objectif. Pour l'exemple considéré, trois personnes différentes ont proposé des solutions différentes aboutissant toutes à l'objectif voulu. La proposition 1 y arrive en seulement trois étapes de calcul, la proposition 2 en cinq étapes, et la proposition 3 en quatre étapes. La seconde étape de la proposition 3 (en caractères italiques) est inutile, mais tout à fait admissible puisque le résultat final est 902. Ces trois propositions ne sont qu'un échantillon des propositions possibles : elles sont en fait au nombre de 204.

Proposition 1 :
8 + 10 = 18
50 × 18 = 900
900 + 2 = 902

Proposition 2 :
50 ÷ 5 = 10
10 × 10 = 100
8 + 1 = 9
9 × 100 = 900
900 + 2 = 902

Proposition 3 :
8 + 10 = 18
18 ÷ 1 = 18
50 × 18 = 900
900 + 2 = 902

Figure 1 : trois exemples de résolution d'un compte

3. Cahier des charges

3.1. Objectif du projet

Le programme Python que vous devez élaborer doit, à partir d'une liste de nombres entiers et d'un nombre entier servant d'objectif, afficher *toutes* les combinaisons possibles menant à cet objectif, *y compris celles contenant des opérations intermédiaires inutiles* (e.g. étape 2 de la proposition 3 de la Figure 1).

3.2. Exemple d'utilisation

Le cœur du programme à développer est la fonction `leCompteEstBon`. À partir d'un programme principal, l'appel `leCompteEstBon([8, 2, 50, 1, 10, 5], 902, "")` doit afficher en retour toutes les combinaisons basées sur la liste `[8, 2, 50, 1, 10, 5]` qui permettent d'aboutir à l'objectif `902` :

```
1  8 * 50 = 400 ; 1 + 10 = 11 ; 400 / 5 = 80 ; 2 + 80 = 82 ; 11 * 82 = 902 ;
2  8 * 50 = 400 ; 1 + 400 = 401 ; 10 * 5 = 50 ; 401 + 50 = 451 ; 2 * 451 = 902 ;
3  8 * 50 = 400 ; 10 * 5 = 50 ; 1 + 400 = 401 ; 50 + 401 = 451 ; 2 * 451 = 902 ;
4  8 * 50 = 400 ; 10 * 5 = 50 ; 1 + 50 = 51 ; 400 + 51 = 451 ; 2 * 451 = 902 ;
5  8 * 50 = 400 ; 10 * 5 = 50 ; 400 + 50 = 450 ; 1 + 450 = 451 ; 2 * 451 = 902 ;
6  8 * 50 = 400 ; 400 / 5 = 80 ; 2 + 80 = 82 ; 1 + 10 = 11 ; 82 * 11 = 902 ;
7  8 * 50 = 400 ; 400 / 5 = 80 ; 1 + 10 = 11 ; 2 + 80 = 82 ; 11 * 82 = 902 ;
8  8 + 1 = 9 ; 2 * 50 = 100 ; 10 / 5 = 2 ; 9 * 100 = 900 ; 2 + 900 = 902 ;
9  8 + 1 = 9 ; 2 * 50 = 100 ; 9 * 100 = 900 ; 10 / 5 = 2 ; 900 + 2 = 902 ;
10 (...)
11 10 / 5 = 2 ; 8 + 1 = 9 ; 2 * 9 = 18 ; 50 * 18 = 900 ; 2 + 900 = 902 ;
12 10 / 5 = 2 ; 2 * 50 = 100 ; 8 + 1 = 9 ; 100 * 9 = 900 ; 2 + 900 = 902 ;
13 10 / 5 = 2 ; 50 * 2 = 100 ; 8 + 1 = 9 ; 100 * 9 = 900 ; 2 + 900 = 902 ;
```

3.3. Généralités sur la fonction `leCompteEstBon`

La fonction `leCompteEstBon` aura le prototype suivant : `leCompteEstBon(liste, objectif, chaine)` :

- l'argument `liste` correspond à la liste de nombres entiers utilisés pour atteindre l'objectif. Les règles officielles du jeu restreignent ce nombre à 6 nombres. Néanmoins, votre fonction devra fonctionner pour une *liste contenant un nombre quelconque d'entiers* ;
- l'argument `objectif` correspond au nombre entier à atteindre ;
- l'argument `chaine` est une chaîne de caractères stockant les étapes des calculs qui ont été précédemment effectués.

L'algorithme développé dans le cadre de ce projet sera un algorithme de type *recherche par force brute*, aussi appelé *recherche exhaustive*. Il s'agit, pour un problème donné, d'explorer l'ensemble des solutions possibles, et de conserver uniquement celles qui répondent à une condition donnée. Dans le cadre de ce projet, les solutions à conserver sont celles pour lesquelles le résultat final est égal à l'objectif `objectif`.

3.4. Fonctionnement imposé pour la fonction `leCompteEstBon`

Le programme principal final se résumera aux trois lignes suivantes :

```
1  testList = [8, 2, 50, 1, 10, 5]
2  testTarget = 902
3  leCompteEstBon(testList, testTarget, "")
```

Pour tester votre programme, les lignes 1 et 2 du programme principal ci-dessus pourront, et devront, évidemment être modifiées. Pour mener à bien l'élaboration de la fonction `leCompteEstBon`, on suivra les étapes ci-dessous.

- **Étape 1 :** définir la fonction `getPermutationsPythonList` ci-dessous. Cette fonction nécessite l'importation préalable du module `itertools`.

```
1  def getPermutationsPythonList(list1):
2
3      list2= range(len(list1))
4      return list(itertools.combinations(list2, 2))
```

Cette fonction retourne toutes les combinaisons de couples d'indices possibles pour une liste donnée, sans notion d'ordre. Par exemple, `getPermutationsPythonList([8, 2, 50, 1])` retournera la liste : `[(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)]`.

- **Étape 2 :** définir une fonction `popList` de prototype `popList(consideredList, index1, index2)`. Cette fonction doit, à partir de la liste `consideredList`, retourner une nouvelle qui ne possède pas les éléments présents aux indices `index1` et `index2` de la liste `consideredList`. Par exemple, l'appel suivant : `popList([8, 2, 50, 1, 10, 5], 1, 3)` devra renvoyer la liste : `[8, 50, 10, 5]`.

- **Étape 3 :** définition de la fonction `leCompteEstBon(liste, objectif, chaine)` :

Un appel à la fonction `leCompteEstBon` doit enclencher le processus de recherche exhaustive de toutes les solutions au compte considéré. En pratique, cette fonction mettra en jeu des séries d'appels récursifs. Pour chaque couple de nombres de la liste `liste`, la fonction `leCompteEstBon` devra lancer des appels à elle même en remplaçant les valeurs de la liste `liste` par leur somme, leur multiplication, leur soustraction et leur division. Ces opérations doivent s'effectuer en respectant les contraintes imposées par les règles du jeu. La Figure 2 illustre le fonctionnement désiré pour la fonction `leCompteEstBon`. On obtient une structure en *arbre*, avec de multiples *branches*.

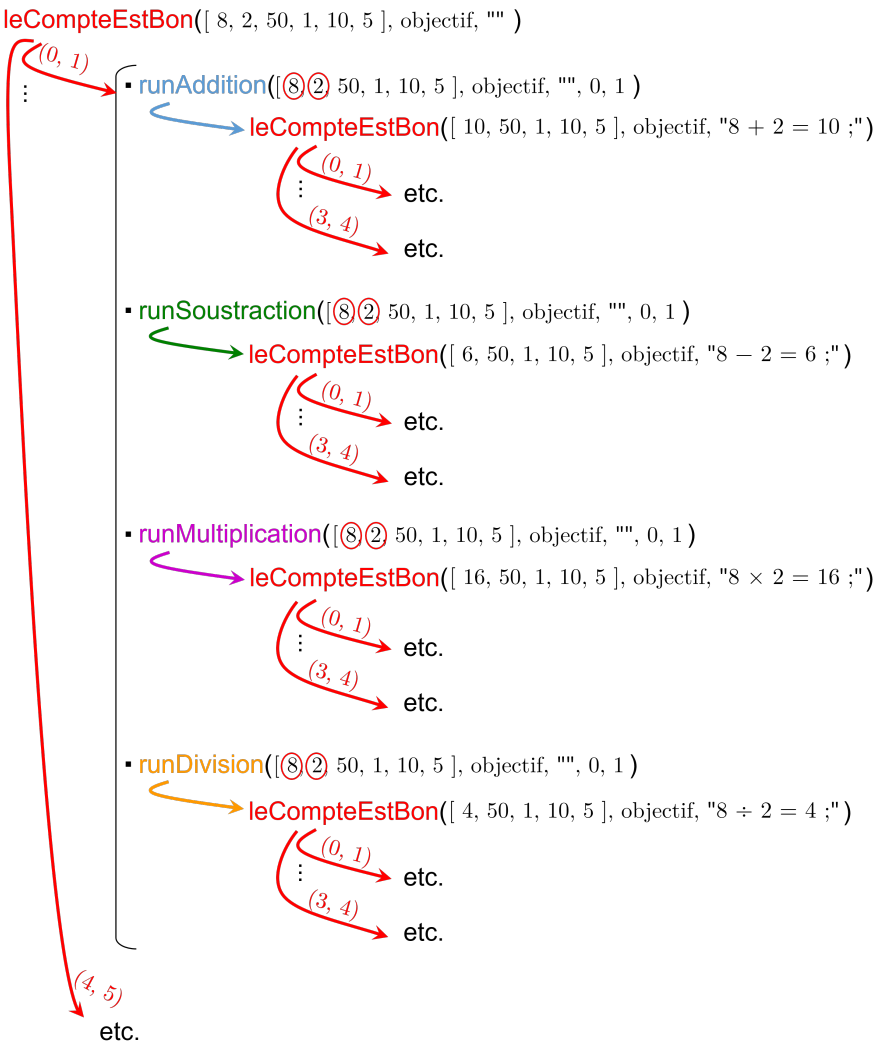

Figure 2 : schéma du fonctionnement de la fonction `leCompteEstBon`

Les appels récursifs à la fonction `leCompteEstBon` seront réalisés indirectement, par l'intermédiaire de quatre fonctions auxiliaires `runAddition`, `runSoustraction`, `runMultiplication` et `runDivision`. Ces quatre fonctions prennent comme arguments : la liste d'éléments courante considérée, l'objectif à atteindre, la chaîne de caractères correspondant aux opérations déjà effectuées, ainsi que les deux indices des deux nombres à combiner par addition, soustraction, multiplication ou par division. Chacune de ces fonctions crée une nouvelle liste à partir des deux nombres combinés et des nombres restants, puis fait appel à la fonction `leCompteEstBon` en utilisant cette nouvelle liste. L'opération effectuée est concaténée à la chaîne de caractères initiale. Par exemple, si la chaîne initiale est vide (`''`), et que l'on considère la somme de 8 et de 2, alors la nouvelle chaîne de

caractères sera : `‘‘8 + 2 = 10;’’`. Si la chaîne initiale n’est pas vide, (e.g. `‘‘50 + 1 = 51;’’`), alors la nouvelle chaîne sera : `‘‘50 + 1 = 51; 8 + 2 = 10’’`.

Quand une succession d’opérations aura permis d’aboutir à l’objectif, la fonction `leCompteEstBon` sera chargée d’afficher la liste des opérations correspondantes, contenue dans la chaîne de caractères `chaine`.

3.5. Autres contraintes

Le programme développé devra respecter les contraintes suivantes :

- être développé en langage Python (version 3) ;
- n’utiliser comme module additionnel que le module `itertools`. De plus, les fonctionnalités de ce module ne pourront pas être utilisées ailleurs qu’au sein de la fonction `getPermutationsPythonList`, fournie dans cet énoncé. Toute modification de la définition de cette fonction est, par ailleurs, interdite ;
- les comptes comportant des étapes intermédiaires inutiles seront acceptés comme solution valable. Par contre, les comptes qui comprendraient des étapes supplémentaires une fois que l’objectif a été atteint ne seront pas considérés comme valables.

4. Question sur le fonctionnement du programme

Pour une liste initiale de six nombres, donner une approximation du nombre de solutions recherchées par la fonction `leCompteEstBon`. Expliquer votre raisonnement.