Université Polytechnique des Hauts-De-France
Institut des Sciences et Techniques ISTV

Licence 1 : TP d'Algorithmique Semestre2 - 2019
Résolution du bon compte

L'objectif de ce TP est de de créer un programme qui permet de trouver , à partir de six nombres entiers, et des 4 opérations entières, les étapes de calculs permettant d'atteindre un nombre objectif.
Un nombre ne peut être utilisé qu'une seule fois.
Exemple,

```
A partir des opérations sur les nombres  [10  5 75  4  6  1]
le but est d'obtenir  654
--
Fin de recherche, valeur trouvée =  654
Nb d'essais effectues =  2580
Détail des opérations :
75 + 10 = 85
5 + 1 = 6
4 * 6 = 24
24 + 85 = 109
6 * 109 = 654
```

Ce TP utilise un type particulier de recherche de solution qui est la recherche aléatoire (ou méthode de Monte-Carlo).
A chaque étape, on prends 2 nombres au hasard (**a** et **b**) parmi les restants (2 parmi 6 au début), on sélectionne aléatoirement une opération **op** compatible; c'est à dire que **a op b** doit donner un nombre entier strictement positif. *Par exemple, si **a=5** et **b=75**, les opération **-** et **/** ne sont pas valides, seules **+** ou **x** peuvent être sélectionnées.* (**a** et **b**) sont extraits du tableau, le résultat de **a op b** est ajouté au tableau.
On répète 5 fois ce processus.. Par exemple, cela peut donner :

```
Etape 1
tab = [10, 5, 75, 4, 6, 1]
tirage de a et b (a=5, b=75) —> tab = [10, 1, 6, 4, 0, 0],
tirage de op (+) ->  tab = [10  4  6  1 80 0],
Etape 2
tab = [10, 1, 6, 4, 80, 0]
tirage de a et b (a=80, b=6) —> tab = [10, 1, 4, 0, 0, 0],
tirage de op (-) ->  tab = [10, 1, 4, 74, 0, 0],
Etape 3
tab = [10, 1, 4, 74, 0, 0]
tirage de a et b (a=10, b=4) —> tab = [74, 1, 0, 0, 0, 0],
tirage de op (x) ->  tab = [74, 1, 40, 0, 0, 0],
Etape 4
tab = = [74, 1, 40, 0, 0, 0]
tirage de a et b (a=40, b=1) —> tab = [74, 0, 0, 0, 0, 0],
tirage de op (-) ->  tab = [74, 0, 0, 0, 0, 0],
Etape 5
tab = [74, 0, 0, 0, 0, 0]
tirage de a et b (a=74, b=39) —> tab = [0, 0, 0, 0, 0, 0],
tirage de op (x) ->  tab = [2886, 0, 0, 0, 0, 0],
```

On s'arrête donc au pire au bout de 5 étapes, ou dès que le but est trouvé (ici 654)
L'objectif n'étant pas atteint, on réitère l'ensemble de l'algorithme précédent; tant que la solution n'est pas trouvée (ce qui suppose qu'il y en a forcément une) et.ou tant que le nombre d'essais max n'est pas atteint.

On utilisera une structure Calcul qui contient la valeur obtenue suite à un essai de résolution, ainsi que la suite des étapes effectuées pour obtenir cette valeur.
On placera en tête du fichier python :

```
from numpy import  array,zeros
from typing import Iterable
from random import randint

##Les Types
TabInt = Iterable[int]
TexteOpe = (str,15)
TabOpe = Iterable[str]

class Calcul:
    etapes:TabOpe=None
    valeur:int=0

##Les constantes
SIGNES:TabOpe=array(["+", "*", "-", "//"])
```

---

1. Définir la procédure **`copier_tab(src:TabInt, dest:TabInt, nb:int)`** qui copie le contenu des **`nb`** cases du tableau **`src`** dans le tableau **`dest`**

---

2. Définir la fonction **`extraire(tab:TabInt, i:int, pos_fin:int)->int`** qui retourne la **`i`**ème valeur de **`tab`**, et la supprime du tableau. Au plus simple, elle remplace cette **`i`**ème valeur par celle en position finale, et place un zéro en position finale.
   Exemple si `tab=[6,5,4,3,2,0]`, ***`extraire(tab, 2, 4)`*** *retourne 4*
   et modifie `tab` *qui devient* `tab=[6,5,2,3,0,0]`

---

3. Définir la fonction **`verifier(op:str, a:int, b:int)->bool`** qui vérifie que l'opérateur **`op`** sur **`a`** et **`b`** est faisable ou utile en respectant les contraintes suivantes :
   1. Si `op="/"` : si division par 1, ou résultat non entier, retourner `False`
   2. Si `op="*"` : si multiplication par 1, retourner `False`
   3. Si `op="-"`, si résultat négatif, retourner `False`
   4. Dans tous les autres cas, la procédure retourne `True`

---

4. Définir la fonction **`calculer(op:str, a:int, b:int)->int`** qui retourne le résultat entier de l'opération **`a op b`** (`calculer("+", 4, 5)` retourne `9`, …, `calculer("/", 984, 8)` retourne `123`).

---

5. Définir la fonction **`choisir_operateur(a:int, b:int)->str`** qui pioche un opérateur applicable sur a et b (vérifié par la fonction précédente) et le retourne.
   La fonction python **`randint(low,top)`** retourne un entier pris au hasard entre **`low`** et **`top`** inclus.
   Il s'agit donc de piocher un nombre `x` en 0 et 3, de récupérer le signe correspondant (`SIGNES[x]`), et de vérifier que le signe est compatible avec `a` et `b` avant de le retourner.

---

6. On dispose maintenant d'une fonction permettant d'extraire des valeurs d'un tableau, d'une fonction permettant de choisir au hasard une opération compatible avec les valeurs, d'une fonction permettant de calculer le résultat de l'opération sur les valeurs extraites.
   Il « ne reste plus qu'à » rédiger en python l'algorithme qui permet un essai de résolution.
   Cet algorithme est le suivant :

tab_num est le tableau d'entiers contenant les nombres de base
but est le nombre entier à atteindre

```
Fonction essayer_calcul(tab_num:TabInt, but:entier):Calcul
     var calcul:Calcul ← nouveau Calcul()
     var copie_num:TabInt ← nouveau tableau de 6 cases de type entier
     var j:entier ← 0, a:entier ← 0, b:entier ← 0
     var signe:char ← ""
Debut:
     copier_tab(tab_num, copie_num, 6)
     calcul.etapes ← nouveau tableau de 5 cases de type TexteOpe
     Pour nb ←5 à 1 Faire
         //prendre une valeur entre 0 et nb
         i ← randint(0, nb)
         a ← extraire(copie_num, i, nb)
         //prendre une valeur entre 0 et nb-1
          i ← randint(0, nb-1)
          b ← extraire(copie_num, i, nb-1)
         //choisir au hasard un signe compatible avec a et b
         signe ← choisir_operateur(a, b)
         //calculer le résultat de l'opération et le ranger en nb-1
         valeur ← calculer(signe, a, b)
         copie_num[nb-1] ← valeur
         calcul.valeur ← valeur
         //stocker l'étape dans le tableau de l'objet calcul
         calcul.etapes[j] ← "" + a + signe + b + "=" + valeur
         j ← j+1
         //si le but est atteint, sortir tout de suite
        Si (valeur=but) Alors retourner calcul
    FinPour
    retourner calcul
Fin
```

Traduisez cet algorithme en une fonction Python **`essayer_calcul(tab_num:TabInt, but:int)->Calcul`**

---

7. Bien sûr il est très peu probable que la solution soit trouvée en un seul essai. Il faut donc lancer plusieurs fois la fonction `essayer_calcul(tab_num:TabInt, but:int)->Calcul` tant que le but n'est pas atteint ou tant que le nombre d'essais n'atteint pas un seuil défini.

   Définissez la procédure **`lancer_essais()`** qui définit le but à atteindre, les nombres à utiliser et lance plusieurs fois `essayer_calcul` tant que l'objet calcul retourné ne possède pas la même valeur que celle du but ou tant que le nombre d'essais maximum (100000 par exemple) n'est pas atteint.

   Bonus. Lorsque le nombre d'essais maximum est atteint sans que la solution soit trouvée, afficher la meilleure des solutions (celle dont la valeur trouvée était la plus proche du but recherché).

Testez avec différents tirages.
Exemple : [1,10,4,5,3,50] avec but = 668
Exemple : [1,2,4,8,10,25] avec but = 789
……

Université Polytechnique des Hauts-De-France
Institut des Sciences et Techniques ISTV